HALO : HOP-BY-HOP ADAPTIVE LINK-STATE OPTIMAL ROUTING

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Nithin Michael

August 2013

HALO : HOP-BY-HOP ADAPTIVE LINK-STATE OPTIMAL ROUTING

Nithin Michael, Ph.D.

Cornell University 2013

Current intra-domain routing protocols like OSPF and IS-IS use link-state routing algorithms with hop-by-hop forwarding that sacrifice traffic engineering performance for ease of implementation and management. Though optimal traffic engineering routing algorithms exist, they tend to be either not link-state algorithms or to require source routing – characteristics that make them difficult to implement. As the focus of this dissertation, we introduce HALO, the first optimal link-state routing algorithm with hop-by-hop forwarding. Furthermore, we will show that our solution can adapt to changing network topology and traffic patterns automatically because the link weights that the protocol uses are purely based on link rates – a useful property in dynamic network conditions. The optimality of the algorithm is proved, its search trajectory is compared with previously proposed optimal routing schemes and the affects of its co-existence with single path routing is studied. Additionally, we present the results of numerical evaluations on several test cases including the benchmark Abilene network that confirms the optimality of the algorithm, evaluates its speed of convergence and adaptivity under changing network conditions and provides evidence of the ability of the protocol to handle asynchrony in the network. We also present results from an experimental testbed that provides further evidence of the optimality, rate of convergence and adaptivity of the HALO protocol.

BIOGRAPHICAL SKETCH

Nithin Michael received the B.S. degree summa cum laude from Drexel University, Philadelphia, PA, in 2008 and the M.S. degree from Cornell University, Ithaca, NY, in 2012, both in electrical engineering. He is currently a Ph.D. candidate in electrical engineering at Cornell University where his research is focused on the optimization of engineering networks. Mr. Michael was the recipient of first Honors in ECE and the Highest Academic Achievement Award at Drexel University in 2008. At Cornell University, he was a Jacobs Fellow in 2008 and 2010.

This document is dedicated to my loving family - I wouldn't be here without you.

ACKNOWLEDGEMENTS

First, I would like to thank my adviser, Dr. Ao (Kevin) Tang. Over the last five years, he has been my guide, my friend and as we sometimes joke amongst ourselves my cheerleader! Graduate school has been one of the most important periods in my development as an individual both intellectually and emotionally. Needless to say, Kevin's presence as a role model for excellence and stoicism has been a significant factor during this journey. As this phase of my life draws to a close, I am lucky to say that I am excited and look forward to continuing to work with Kevin as we embark on a new adventure.

I was very fortunate that Dr. Lang Tong showed the willingness to reach out to me when I first began exploring Cornell for graduate school and I am grateful to him for being on my committee from the beginning and for sharing his experience and wisdom freely over the years. I was also lucky to have Dr. Edward Suh on my committee. He has been nothing but encouraging, understanding, patient and supportive over the last several years. Research can be and in my case at least, has been, very frustrating at times. Ed's willingness to always make time for me and to help me find what can be salvaged was such a relief during those times.

I would also like to acknowledge the excellent work done by Ning, Hans, Amrisha and Pranjal in building the testbed that was used to perform the experimental work reported in this dissertation.

I wouldn't be anywhere without my family - Daddy, Mommy, Nevin and Ammamma. I love them and wouldn't know what to do without their presence, their understanding and their love. It has carried me through even the most difficult challenges.

Ithaca has been a wonderful experience for me even when research was at its most challenging. For this I am grateful to my friends Gyanam, Mihail, Enrique,

Meng, Frank, Wacek, Caroline, Carlos, Milen, Chiunlin, Yucel, Ali, Ilan, Shiyao, Rajeev, Xavier, Father Dan, and everyone else that has made the past five years so memorable.

Lastly, I would like to thank Barbara for her kindness, her support, her encouragement and her love. All I can say is that it was really worth the wait!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Finding optimal routes [1] in packet-switched networks has been of fundamental research and practical interest since the early 1970s with the advent of ARPANET [2], the predecessor of the Internet. But today, sub-optimal distributed link-state routing protocols with hop-by-hop forwarding like OSPF and IS-IS are the dominant intra-domain routing solutions on the Internet. As the network scaled, it was clear that the simplicity of these schemes (the main idea is to centrally assign appropriate weights to links and compute shortest paths using Dijkstra's algorithm) made them easier to implement and manage compared to the optimal solutions that had been proposed. Therefore, these algorithms have become ubiquitous despite their potentially very large performance loss. Some of the lost performance was recouped through extensive capital expenditure. For instance, due to the poor resource utilization resulting from these protocols, many links of the internet are so over-provisioned to support peak traffic that they run at very low utilizations on average. Unsurprisingly, the search for an optimal routing algorithm that has the same ease of management and implementation as OSPF has continued unabated.

1.1 Motivation

Our work was motivated by the performance loss of OSPF and IS-IS and the resulting inefficiencies. In fact, given the offered traffic, finding the optimal link weights, if they even exist, for OSPF/IS-IS is a well-known NP-hard problem [3]. Furthermore, it is possible for even the best weight setting to lead to traffic that deviates significantly from the optimal traffic distribution [3].

But before we proceed, we define a few basic terms followed by some more background to motivate our study.

Link-state: Routers make routing decisions based on knowledge of the network topology and the weights associated with the links.

Hop-by-hop: Each router, based on the destination address, controls only the next hop that a packet takes.

Optimal: The routing algorithm minimizes some cost function (e.g. minimize total delay) determined by the network operator. The problem of guiding network traffic through routing to minimize a given global cost function is called traffic engineering (TE).

Adaptive: the algorithm does not require the traffic demand matrix as an explicit input in order to compute link weights. Specifically, the algorithm seamlessly recognizes and adapts to changes in the network, both topology changes and traffic variations, as inferred from the network states like link flow rates.

As can be expected, designing an optimal protocol while keeping the simplicity of link-state hop-by-hop protocols comes with a few challenges. Firstly, relying only on link-state information means that no router is aware of the individual communicating pairs in the network or their requirements and yet have to act independently such that the TE objective is optimized. This is a very real restriction as in any large dynamic network like the Internet, it is not possible to obtain information about individual communicating pairs. If the link-state requirement is set aside, optimal distance-vector protocols that rely on locally transmitted node-states exist [4]. However, the main reason that a distance-vector protocol is not preferred for intra-domain routing is because it suffers from scalability issues as

well as decreased robustness like vulnerability to a single rogue router taking down the network as in the "Internet Routing Black Hole" incident of 1997 [5].

Secondly, the hop-by-hop forwarding requirement prevents routers from controlling anything except the next hop that a packet can take. As a result, routers cannot take advantage of any additional information that they have about traffic originating with them. If this requirement is set aside, a projected gradient approach [6] can be used to yield optimal link-state algorithms which can be implemented with source routing, where the path a packet takes through the network is encoded in its entirety at the source. Even though such schemes can be implemented with MPLS [7, 8], optimality comes at the cost of establishing multiple end-to-end virtual circuits. Moreover, as the traffic changes, the end-to-end virtual circuits that were established for a particular traffic pattern become less useful and performance degrades.

Lastly, our solution has the advantage of being adaptive and if this requirement is set aside, recently significant progress was made in this direction with PEFT, a link-state protocol with hop-by-hop forwarding based on centralized weight calculations [9]. Since the link weights are calculated in a centralized manner with the traffic matrix as an explicit input, PEFT is not adaptive. Also, PEFT does not always guarantee optimality as claimed in the paper[1].

[1]When the optimal routing solution does not use all available paths to the destination, as is the case in many networks, for example, any network with at least a loop, a set of finite optimal weights for PEFT [9] does not exist. For more details please see Appendix A.

1.2 Contributions

In this dissertation, we present HALO (Hop-by-hop Adaptive Link-state Optimal), which to the best of our knowledge, is the first optimal link-state hop-by-hop routing algorithm. The main contribution of our work is the design of the HALO protocol and the proof of its optimality. We also study how the search trajectories followed by the different protocols that have been proposed for optimal traffic engineering are different and construct an example where each protocol calculates a different optimal solution. Additionally, we explore how HALO can co-exist with a single path routing protocol in a network and demonstrate how the protocol strictly improves network performance in such a situation. We support our theoretical arguments through numerical evaluations on several test cases including the benchmark Abilene network. The results verify the optimality and the adaptivity of the protocol and provide insight into its rate of convergence. We also report the results of experiments on a real network testbed running the HALO protocol that further verify our predictions about the optimality and adaptivity of HALO.

The dissertation is organized as follows. In Chapter 2 we review several different solutions that have been proposed for the traffic engineering problem and clearly identify the missing piece that is provided by our work. Next we present the Multicommodity Flow problem which provides the model for our analysis and introduce notation in Chapter 3 before presenting and analyzing our solution in Chapters 4 and 5, followed by further discussion in Chapter 6. Numerical and experimental evaluations are used to verfiy the performance of the protocol in Chapter 7 and Chapter 8 before we conclude the dissertation with a summary and future research directions in Chapter 9.

CHAPTER 2

BACKGROUND ON ROUTING IN INFORMATION NETWORKS

Over the years, due to its importance, the TE problem has attracted a lot of research attention from different research communities. Below we provide a brief overview of major related existing results to show where our work fits. While no means exhaustive, since this review includes papers from different communities such as control, optimization, networking, and theoretical computer science, we hope that this chapter can help researchers interested in this topic find important results from other communities.

Broadly, the existing work can be divided into studies of heuristic improvements to OSPF, designing traffic demand agnostic/oblivious routing protocols and studies of optimal routing algorithms. The work on OSPF [3, 10, 11] was motivated by the need to improve its efficiency after it had become the dominant routing protocol on the internet. However, though these techniques have been shown to improve the performance of OSPF significantly by finding better weight settings for the algorithm, the results are still far from optimal. Typically, these efforts also assume that a good estimate of the traffic demand in the form of a traffic matrix is available. While excellent work has been done in the direction of traffic matrix estimation, even the best results have errors in the elements of the estimated traffic matrix on the order of 20% [12] – difficulties which can lead to potentially bad traffic engineering. As Figure 2.1 illustrates, we are proposing a shift in how traffic engineering is performed by advocating a move to relying directly on the link loads to track the optimal routing solution.

Oblivious routing has been proposed as a way around the need for estimating the traffic matrix for good TE. The idea is to come up with a routing solution

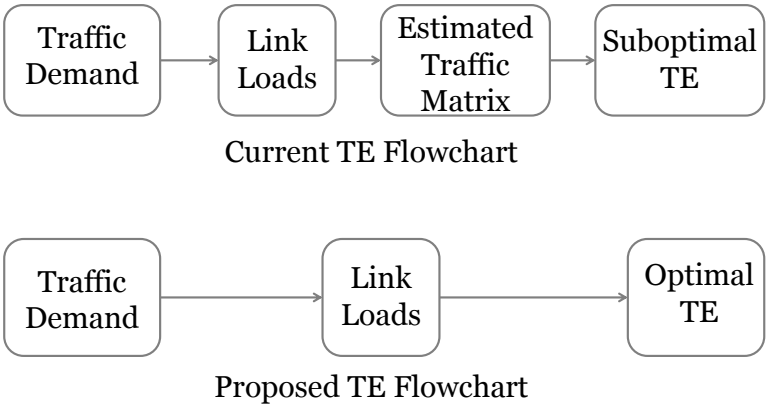


Figure 2.1: Flowcharts illustrating the difference between TE today and the proposed technique.

that performs well irrespective of the traffic demand by comparing the 'oblivious performance ratio' of the routing, i.e., the worst case performance of the routing for a given network over all possible demands. Breakthrough work in this area include papers by Applegate and Cohen [13] that developed a linear programming method to determine the best oblivious routing solution for the special case of minimizing maximum channel utilization and Kodialam et al [14] that focused on maximizing throughput for the special case of two phase routing. But some important drawbacks of these methods are that the oblivious routing solutions do not adapt well to changes in the network topology and that by not taking advantage of actual traffic information the routing still incurs significant performance losses.

On the other hand, there are optimal solution techniques that deliver on performance without the need for an estimated traffic matrix by relying on knowledge of the link flow rates. They can be broadly classified into protocols that are distributed per node and protocols that are distributed per commodity or source-

destination pair as these approaches are representative of the main ideas used to tackle the optimal routing problem.

The class of decentralized algorithms that are node-based are distance-vector protocols and include the ones proposed by Gallager in his classic paper [4], Stern [15] and Agnew [16]. The idea behind these algorithms was that as long as a node was aware of the "price" ("average distance") to each destination at each of its neighbors, it had enough information to make optimal forwarding decisions. From an optimization standpoint, this is a natural and mathematically elegant approach since the main ideas follow directly from the decomposition of the dual of the TE optimization problem. Decompositions like this, which have been very successful for problems of this type [17], can be used to yield updating rules for both primal and dual variables (split ratios and node prices in [4]) that can be shown to converge to optimal solutions. Similar node-based ideas have also been applied to cross-layer optimization of networks [18, 19].

On the other hand, we have optimal link-state routing algorithms that are decentralized by source-destination pairs (path-based). Examples of this variety include the flow deviation technique advocated by Fratta et al [2], projection methods proposed by Bertsekas and Gafni [6] as well as proximal decomposition methods [20]. These solutions are based on iteratively calculating a shortest path at the source for each commodity and transferring varying amounts of flow from the non-shortest paths to the shortest path to obtain the optimal traffic distribution. An important limitation of these algorithms is that, as noted earlier, they require source routing, i.e., the route a packet will take through the network has to be completely encoded at the source.

Another direction of inquiry centered on decentralization by source-destination

Table 2.1: Comparison of Routing Algorithms

Algorithm	**Link-state**	**Hop-by-hop**	**Optimal**	**Adaptive**
OSPF	✓	✓	×	×
Gallager's [4]	×	✓	✓	✓
Projected Gradient [6]	✓	×	✓	✓
PEFT [9]	✓	✓	×	×
HALO	✓	✓	✓	✓

pairs, and thus requiring source routing, focused on the development of approximation algorithms that had bounded computational requirements. Many excellent papers were written in this area [21, 22, 23, 24], with the guarantees on the computational requirements improving steadily over the years. While in this dissertation we do not provide theoretical analysis on the run-time of our algorithm, our algorithm just adds simple operations on top of standard shortest path algorithms and our numerical evaluations show that even for large networks our algorithm converges fairly quickly to an optimal solution.

The preceding overview of how the literature has developed in this area, clearly reveals the missing link in the search for an optimal link-state hop-by-hop routing algorithm. In this dissertation, we provide this missing link by introducing HALO. A comparison of several existing solutions and ours can be seen in Table 2.1.

CHAPTER 3

MULTICOMMODITY FLOW AND OPTIMAL ROUTING

The optimization problem that is used for traffic engineering is the Multi-Commodity Flow problem (MCF). For a given directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with node/router set $\mathbb{V}$ and edge/link set $\mathbb{E}$ with link capacities $c_{u,v},\ \forall (u,v) \in \mathbb{E}$, and demands $D(s,t)$ defined as the rate required for communication from s to t, the MCF problem can been summarized below.

$$\min_{f^t_{u,v}} \ \Phi(f)$$

$$s.t. \sum_{v:(s,v)\in\mathbb{E}} f^t_{s,v} - \sum_{u:(u,s)\in\mathbb{E}} f^t_{u,s} = D(s,t),\ \forall s \neq t$$

$$f_{u,v} = \sum_{t\in\mathbb{V}} f^t_{u,v} \leq c_{u,v},\ \forall (u,v)$$

$$f^t_{u,v} \geq 0$$

Here commodities are defined in terms of their final destination t. $f^t_{u,v}$ is the flow on link (u,v) corresponding to commodity t and $f_{u,v}$ is the total flow on link (u,v). The cost function, Φ, is typically selected to be a convex function of the link rate vector $f = \{f_{u,v}\},\ \forall (u,v) \in \mathbb{E}$. For example, if we use the M/M/1 delay formula for the cost function, then $\Phi(f) = \sum_{u,v} \Phi_{u,v}(f_{u,v}) = \sum_{u,v} f_{u,v}/(c_{u,v} - f_{u,v})$ [1]. Throughout the dissertation, for numerical examples, we will use this cost function unless specified otherwise. It is also assumed that $\Phi'_{u,v}(f_{u,v}) \to \infty$ when $f_{u,v} \to c_{u,v}$. This captures the common practice of not allowing links to operate too close to capacity. In this dissertation, given a function $\gamma(x(\tau))$, we will use γ' to represent the derivative of γ with respect to x and $\dot{\gamma}$ to represent the time (τ) derivative of γ.

Provided that the objective does not have a barrier to approaching capacity built into it, we can modify it slightly as is commonly done in the literature as well as in implementation. The trick, as illustrated in Figure 3.1, is to charge a very high cost as a link approaches capacity which is something that we wish to do anyway.

We also define the price of a link (u, v) as $w_{u,v} = \Phi'_{u,v}(f_{u,v})$, the price of a path p as $\sum_{(u,v)\in p} w_{u,v}$ and the price at a node u to a destination t as,

$$q_u^t = \sum_{v:(u,v)\in\mathbb{E}} \alpha_{u,v}^t [w_{u,v} + q_v^t] \tag{3.1}$$

where $q_t^t = 0$. The price at a node can be interpreted as the average price to the destination from that node where the average is taken over all outgoing edges to the destination weighted by the split ratios along those edges. If instead the average is done over all possible paths, Equation (3.1) can be stated without recursion as,

$$q_u^t = \sum_{p\in P_{u,t}} d_p \prod_{(i,j)\in p} \alpha_{i,j}^t \tag{3.2}$$

where $P_{u,t}$ is the set of all paths from u to t and $d_p = \sum_{(u,v)\in p} w_{u,v}$.

A fact about MCF is that its optimal solution generally results in multi-path routing instead of single-path routing [25]. However, finding the right split ratios for each router for each commodity is a difficult task. Our starting point is to merge the link-state feature of protocols that were decentralized by source-destination pairs with the hop-by-hop forwarding feature of the node-based decentralization schemes. More concretely we,

- adjust each router's split ratios and move traffic from one outgoing link to another. This only controls the next hop on a packet's path leading to hop-by-hop routing. If instead we controlled path rates we would get source routing.

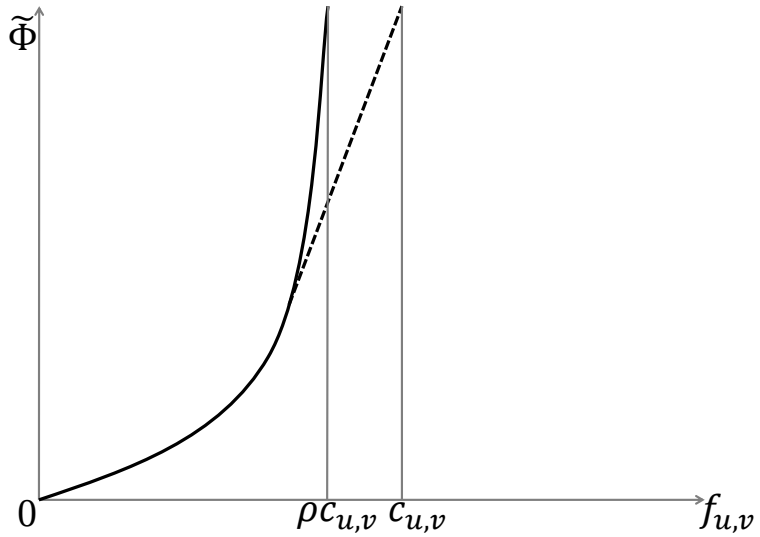


Figure 3.1: **Modifying** Φ**.** Make the cost increase rapidly as capacity is approached, say from a point $\rho c_{u,v}$ for $\rho < 1$.

- increase the split ratio to the link which is part of the shortest path even though the average price via the next hop node may not be the lowest. If instead we forwarded traffic via the next hop node with the lowest average price we get Gallager's approach, which is a distance vector solution.
- adapt split ratios dynamically and incrementally by decreasing along links that belong to non-shortest paths while increasing along the link that is part of the shortest path at every router. If instead split ratios are set to be positive instantaneously only to the links leading to shortest paths, then we get OSPF with weights, $w_{u,v}$.

CHAPTER 4

CASE STUDIES ON OPTIMAL ROUTING

> "The only real valuable thing is intuition."
>
> Albert Einstein

In order to develop an intuitive understanding of why our solution takes the form that it does, it is helpful to consider a few concrete special cases first. These four cases, each of which clearly highlights the reason for including a particular factor in our solution, progressively lead us to the final algorithm, which we will prove to always work for any general case in Chapter 5. The calculations in this chapter also give hints to the main idea of the proof.

4.1 Finding the Right Split Dynamically

First let us consider a very simple example illustrated in Figure 4.1. It is worth noting that the KKT optimality conditions [26] of the MCF problem require that at the optimal solution the traffic rate is positive only along paths with the lowest price. In this example, assuming initially $w_l > w_s$, a simple strategy to reach optimality might be to dynamically shift traffic from the more expensive link to the cheaper link at some rate $\delta > 0$ till the prices of the two links become the same. In terms of the change in split ratios at node A this would be equivalent to decreasing α_l and increasing α_s at rate δ/r.

There are two ways to interpret and generalize the intuition gained from this scenario. Both give the same solution for this very simple example but in general

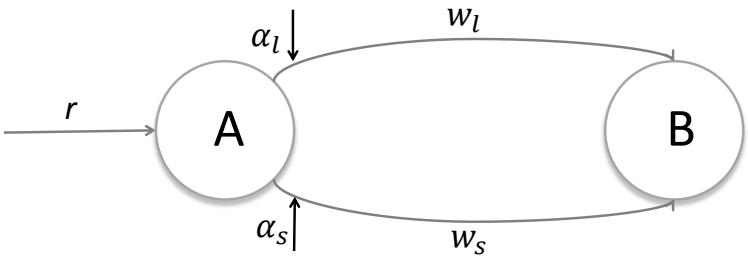


Figure 4.1: **Finding the right split dynamically.** Suppose that there is a single demand of rate r trying to get to destination B. Initially, the split ratios at A are α_l along the more expensive link with price $w_l = \Phi_l'$ and α_s along the cheaper link with price $w_s = \Phi_s'$.

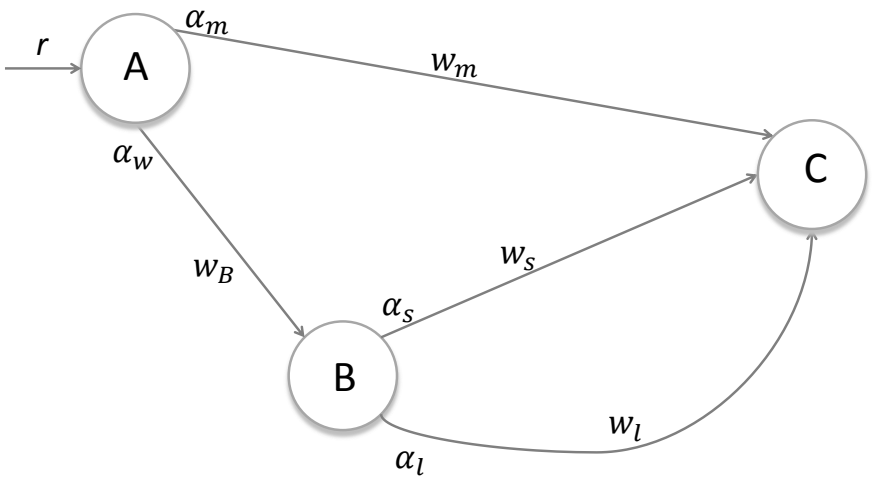


Figure 4.2: **A first test.** Suppose the link weights are as shown and $w_l > w_m > w_s + w_B$. There is a single demand $D(A, C)$.

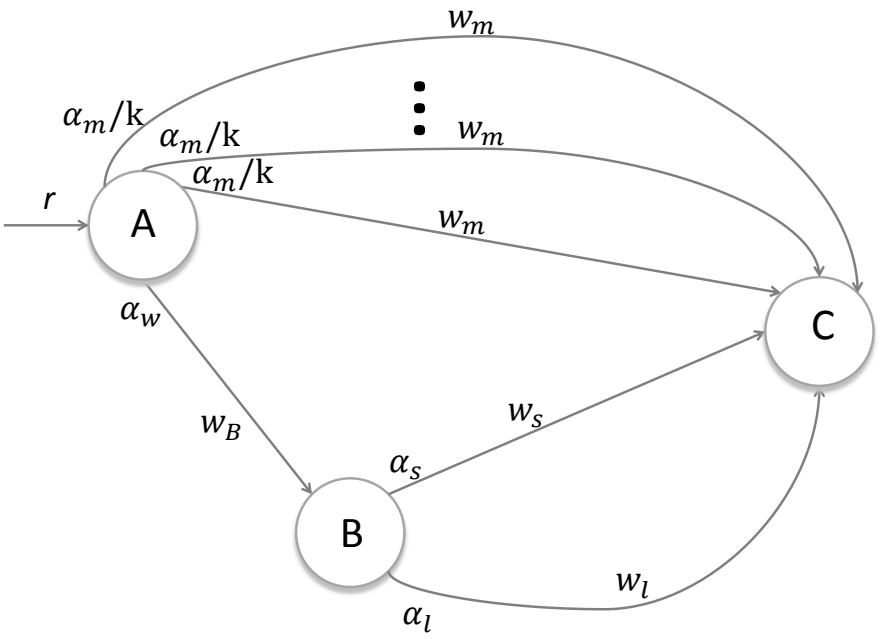


Figure 4.3: **Multiple outgoing paths.** Suppose the link weights are as shown and $w_l > w_m > w_s + w_B$. There is a single demand $D(A, C)$.

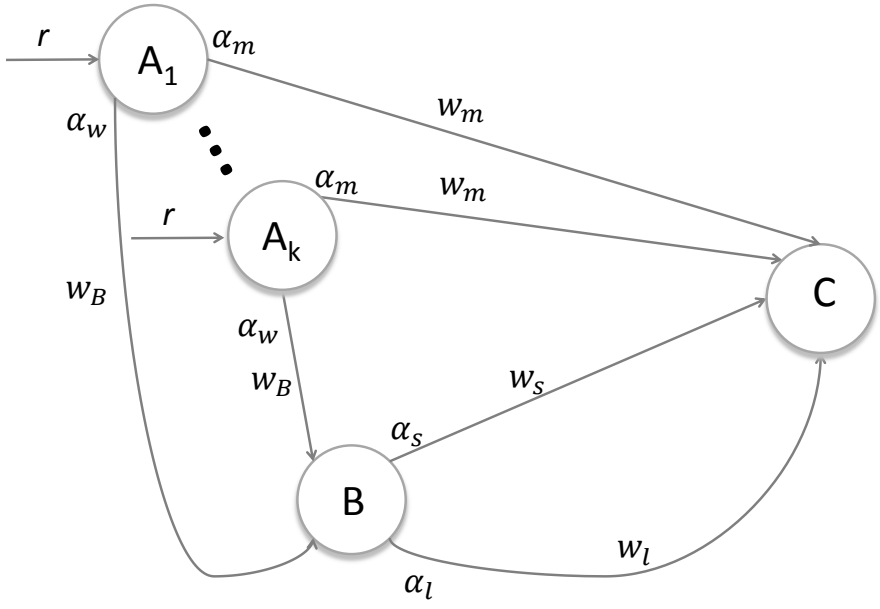


Figure 4.4: **Multiple inputs.** Suppose the link weights are as shown and $w_l > w_m > w_s + w_B$. There are k demands $D(A_i, C),\ i = 1, \ldots, k$.

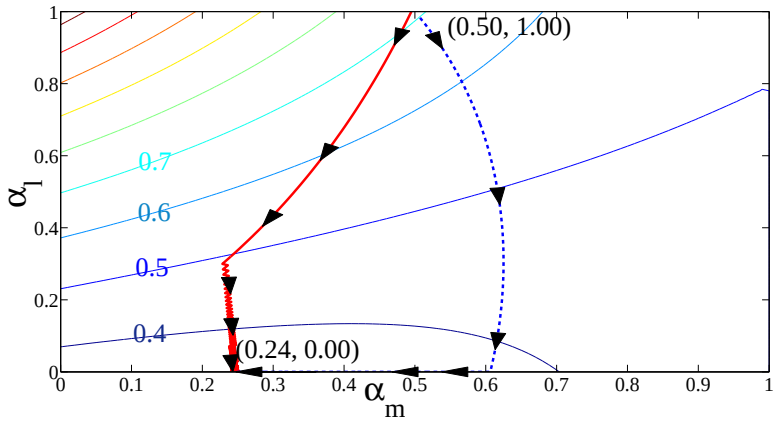


Figure 4.5: Trajectories taken by Gallager's algorithm (dashed line) and our algorithm (solid line) to converge to the optimal solution. Cost values are shown for some contour lines.

will lead to different dynamics (See Figure 4.5) and possibly different split ratios (See Figure 6.1a). One interpretation, which forms the basis of the technique proposed by [4], is that the router shifts traffic headed to neighbor nodes with higher average price to the neighbor node with the lowest average price. A different interpretation, which is the basis of our protocol, is that the router shifts traffic from links along more expensive paths to the link along the path with the lowest price. Mathematically, we reach the following update rule for the split ratios,

$$\dot{\alpha}_{u,v}^{t} = -\frac{\delta}{r_u^t} \tag{4.1}$$

where $(u, v) \in \mathbb{E}$ but is not on the shortest path from u to destination t and r_u^t is the incoming rate at node u to destination t.

4.2 A First Test

However, as a potential counter-example to this interpretation, it is possible to suggest some version of the scenario described in Figure 4.2. Here there is traffic demand of rate r from router A to router C. The initial splits at router A are α_m

along an intermediate price link with price w_m and α_w along the more expensive route with price $w_B + w_l$, assuming $\alpha_l = 1$ initially. The relationship between the initial link prices are assumed to be $w_l > w_m > w_s + w_B$, i.e., link (A, B) is along the shortest path from A to C, but B also has the most expensive way to reach C. The concern is that router A shifting traffic from the intermediate price link to the link with price w_B might result in the cost increasing as router B currently routes traffic only through the most expensive link ($\alpha_l = 1$). But because router B decreases α_l and increases α_s (in conjunction with the changes at router A), the total cost does in fact decrease. More precisely, the cost derative can be calculated as follows,

$$\begin{aligned}\dot{\Phi} =& - r \times \frac{\delta}{r} \times w_m + r \times \frac{\delta}{r} \times (w_B + w_l) \\ & - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\ =& - \delta(w_m - w_B - w_s) \leq 0\end{aligned}$$

where r_B is the incoming rate to C at B (superscript dropped for convenience since C is the only destination) and the inequality follows from the relationship between the prices.

This particular example can also be used to illustrate the difference between our approach and Gallager's technique which arises from the fact that the link leading to the neighbor with the lowest average price (path A-C with price w_m) may not lead to the cheapest path (path A-B-C with price $w_B + w_s$). Figure 4.5 shows the trajectories taken by the two different algorithms to converge to the optimal solution for this topology. In order to simulate the long link between node B and node C, an intermediate dummy node D is introduced that splits the bottom link between B and C into two equal capacity links. The capacities used were $(A, B) = 5, (B, C) = 10, (A, C) = (B, D) = (D, C) = 3$. The rate $r = 1$ and

initially $\alpha_w = \alpha_m = 0.5$ and $\alpha_l = 1$. Here we take only one split ratio at each node because the value of that split ratio automatically defines the value of the other at each node. Using Gallager's algorithm, initially, as can be seen, following the lowest average price path to the destination (A, C), there is an increase in the value of α_m. Also, as expected from theory, the trajectory of the algorithm (gradient descent) is perpendicular to the objective function contour curves. On the other hand, using HALO, both split ratios are decreased initially. HALO's trajectory is usually not perpendicular to the counter curves, however, it still goes along a descent direction and drives the total cost down.

4.3 Multiple Outgoing Paths

The above case study might lead us to ask whether the simple rule developed thus far (Equation (4.1)) is sufficient to guarantee decreasing network cost along any trajectory. In order to see why it is not, consider the situation presented in Figure 4.3. Now there are k intermediate price links from router A to router C each of which gets α_m/k fraction of the demand. The relationship between the link prices is the same as in the previous example. Now the concern is that shifting traffic in an unrestricted fashion from the intermediate price links to router B with $\alpha_l = 1$, might result in an increase in the cost and it is a valid concern as illustrated by the following calculation.

$$\begin{aligned}\dot{\Phi} =& -k \times r \times \frac{\delta}{r} \times w_m + k \times r \times \frac{\delta}{r} \times (w_B + w_l) \\ & - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\ =& -k\delta w_m + \delta(k w_B + w_s) + (k-1)\delta w_l\end{aligned}$$

which may be positive for $k > 1$. But this problem can be surmounted by modifying the update rule followed by the split ratios by adding a weighting factor of the split ratio itself. Mathematically, we have

$$\dot{\alpha}^t_{u,v} = -\frac{\alpha^t_{u,v}\delta}{r^t_u} \tag{4.2}$$

where $(u, v) \in \mathbb{E}$ but is not on the shortest path from u to destination t.

With this new rule, the cost derivative can be evaluated as,

$$\begin{aligned}
\dot{\Phi} =& - k \times r \times \frac{\delta\alpha_m}{rk} \times w_m + kr \times \frac{\delta\alpha_m}{rk} \times (w_B + w_l) \\
& - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\
=& - \delta[\alpha_m w_m + (1 - \alpha_m)(w_B + w_l)] + \delta(w_B + w_l) \\
& - \delta w_l + \delta w_s \\
=& - \delta[\alpha_m w_m + (1 - \alpha_m)(w_B + w_l)] + \delta(w_B + w_s) \\
\leq& 0
\end{aligned}$$

where the last inequality follows from the fact that the average price from router A to C, which is $\alpha_m w_m + (1 - \alpha_m)(w_B + w_l)$ has to be at least as large as the price of the shortest path from A to C, which is $w_B + w_s$.

4.4 Multiple Inputs

Does Equation (4.2) ensure that total cost decreases along its trajectory? Another case worth considering is illustrated in Figure 4.4. This time there are k sources $A_1, \ldots, A_k$ that have data to send to router C. Now the concern is that shifting traffic in an unrestricted manner from all the sources to router B with $\alpha_l = 1$,

could cause the total cost to increase as shown by the calculations below,

$$\begin{aligned}\dot{\Phi} =& - k \times r \times \frac{\delta\alpha_m}{r} \times w_m + k \times r \times \frac{\delta\alpha_m}{r} \times (w_B + w_l) \\ & - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\ =& - k\delta[\alpha_m w_m + (1-\alpha_m)(w_B + w_l)] + (k-1)\delta w_l \\ & + \delta(k w_B + w_s)\end{aligned}$$

which may be positive for $k > 1$. Once again it is possible to modify the update rule for the split ratios from $\delta\alpha^t_{u,v}/r^t_u$ to $\delta\alpha^t_{u,v}/\eta^t_u r^t_u$. In this case, $\eta^t_u = k$ while for a general network we will specify how to calculate η^t_u in Chapter 5, Algorithm 1. With the new rule, the cost derivative will be the same as in Section 4.3, and always no more than zero.

Formally, the above discussion leads us to further modify the update rule in Equation (4.2) to

$$\dot{\alpha}^t_{u,v} = -\frac{\alpha^t_{u,v}\delta}{\eta^t_u r^t_u} \tag{4.3}$$

where $(u, v) \in \mathbb{E}$ but is not on the shortest path from u to destination t. In the following chapter we will show that for any network, this update rule for the split ratios (4.3) makes the total cost of the network always decrease, resulting in the split ratios converging to a set where every element of the set achieves the global optimum to the MCF problem, and therefore achieves optimal TE.

CHAPTER 5

THE HALO PROTOCOL – DESCRIPTION AND ANALYSIS

We first introduce some additional necessary notation. For a particular destination t at node s we define,

$$r_s^t = \sum_{u:(u,s)\in\mathbb{E}} f_{u,s}^t + D(s,t)$$

the inflow rate to a node s destined to t which because of node flow balance requirements is also the outflow at s to t. We will also use α without indexing to represent the set of all the split ratios from all the routers in the network. We have already noted that at a router u, $\alpha_{u,v}^t$ controls the fraction of traffic to destination t that uses outgoing link (u,v) while satisfying $\alpha_{u,v}^t \geq 0$ and $\sum_{v:(u,v)\in\mathbb{E}} \alpha_{u,v}^t = 1$.

Next, we define η_u^t, the *branch cardinality*, as the product of the number of branches encountered in traversing the shortest path tree rooted at t from t to u. Being a link-state routing algorithm, each node u has the link-state information to run Dijkstra's algorithm to compute the shortest path tree to destination t. Here additional care is required because every node has to independently arrive at the same shortest path tree to ensure that the algorithm proceeds as expected. So at any stage of Dijkstra's algorithm, if there is ambiguity as to which node should be added next, tie-breaking based on node index is used. The calculation of η_u^t proceeds as shown in Algorithm 1. For an illustration of how η_u^t is calculated, please refer to the example following the proof of Theorem 1.

5.1 Description

We are now in a position to describe the adaptive link-state routing algorithm. For any node u, it controls the evolution of the destination specific split ratio $\alpha_{u,v}^t$.

Algorithm 1 Algorithm to calculate η_u^t $\{w_e, \forall e \in \mathbb{E}\}$

1: Compute shortest path tree for destination t using Dijkstra's algorithm with tie-breaking based on node index
2: Traverse the tree from t to u
3: Initialize $\eta_u^t \leftarrow 1$
4: At every junction do $\eta_u^t \leftarrow \eta_u^t b$ where b is the number of branches from that junction

Suppose that $(u, \bar{v}) \in \mathbb{E}$ and $(u, \bar{v})$ is part of the shortest path to t from u. Then HALO calculates the split ratios as follows.

$$\text{if } r_u^t > 0, \qquad \dot{\alpha}_{u,v}^t = -\frac{\alpha_{u,v}^t \delta}{\eta_u^t r_u^t}, \ v \neq \bar{v} \tag{5.1}$$

$$\dot{\alpha}_{u,\bar{v}}^t = -\sum_{v:(u,v)\in\mathbb{E},\ v\neq\bar{v}} \dot{\alpha}_{u,v}^t \tag{5.2}$$

$$\text{else if } r_u^t = 0, \qquad \alpha_{u,v}^t = 0, \ v \neq \bar{v} \tag{5.3}$$

$$\alpha_{u,\bar{v}}^t = 1 \tag{5.4}$$

We present the formal description of HALO in Algorithm 2.

Algorithm 2 Forwarding Algorithm at Router u $\{w_e, \forall e \in \mathbb{E}\}$

for all t **do**
    Calculate η_u^t
    if $r_u^t == 0$ **then**
        for all $v \neq \bar{v}$ **do**
            $\alpha_{u,v}^t = 0$
        end for
        $\alpha_{u,\bar{v}}^t = 1$
    else
        for all $v \neq \bar{v}$ **do**
            $\dot{\alpha}_{u,v}^t = -\frac{\alpha_{u,v}^t \delta}{\eta_u^t r_u^t}$
        end for
        $\dot{\alpha}_{u,\bar{v}}^t = -\sum_{v:(u,v)\in\mathbb{E}, v\neq\bar{v}} \dot{\alpha}_{u,v}^t$
    end if
end for

5.2 A Hybrid Systems Perspective

The dynamics of HALO is best studied in the context of hybrid systems theory. Hybrid systems are dynamical systems that have both continuous (the split ratios) and discrete (the current shortest path tree) states. In the analysis that follows, we will invoke a version of the LaSalle Invariance Principle for hybrid systems from Lygeros et al [27]. But in order to be able to apply the theorem, we have to establish that our system is non-blocking, deterministic, continuous and satisfies Assumption II.2 from the paper, which requires that the sets Init and Dom_H are closed and that $\text{Reach}_H = Init \subset \text{Dom}_H$. Here Init is the set of intial states, Reach_H represents the set of states reachable by our hybrid system H and Dom_H represents the domain of H.

We will first argue that our hybrid automaton satisfies Assumption II.2 and then establish that it is deterministic, non-blocking and continuous. For our algorithm, suppose that the set of initial states is any feasible assignment of split ratios. Then $\text{Reach}_H \subseteq \text{Init}$, since all reachable states have to be feasible. Also, any initial feasible state is trivially reachable (by starting from that state) giving us $\text{Init} \subseteq \text{Reach}_H$ and therefore, $\text{Init} = \text{Reach}_H$.

In order to argue that the algorithm is deterministic we will rely on Lemma III.2 [27] which requires a hybrid system to satisfy three conditions for it to be deterministic. The first condition requires that if a discrete transition happens from a state, then continuous evolution should not be possible from that state. In our algorithm, discrete transitions happen when a router changes its shortest path to a destination. At such a state, continuous evolution is indeed impossible since continuous evolution would imply that we did not change the shortest path even though the underlying state had produced a change in the shortest path to the

destination. The second condition requires that when a discrete transition is about to happen, there should be no confusion as to which transition should take place. Because we have imposed tie-breaking between different shortest paths, this will not happen. The last condition requires that when a discrete transition happens, the 'reset' map should have only one element. Since our discrete transitions happen with the split ratios changing continuously across the transition this is true as well. Consequently, our system is a deterministic hybrid automaton.

In order to argue that the system is non-blocking we will rely on Lemma III.1 [27] which essentially states that if we reach a state from which continous evolution is impossible, a discrete transition should be available. For our algorithm this is true by construction.

Lastly, we will argue that our system is a continuous hybrid automaton since it satisfies the definition of continuous hybrid automata from Lygeros et al [27] which basically captures the intuitive idea that for any two states that are close to each other, it should be possible to find two initial states that are close to each other executing from which for approximately the same time should lead to those two states. Since our system is locally Lipschitz continuous in between discrete transitions and because by definition, states separated by discrete transitions cannot be arbitrarily close to each other, it is easy to see why our system satisfies the requirements to be classified as a continuous hybrid system.

5.3 Analysis

To establish that HALO leads to optimal TE, we will need the following two lemmas. The first one which was derived by Gallager [4], relates the node prices

to the link weights for each destination t.

Lemma 1. $\sum_{u\in\mathbb{V}} D(u,t)q_u^t = \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t w_{u,v}$

Proof.

$$\begin{aligned}
\sum_{u\in\mathbb{V}} r_u^t q_u^t &= \sum_{u\in\mathbb{V}/t} r_u^t \sum_{v:(u,v)\in\mathbb{E}} \alpha_{u,v}^t [w_{u,v} + q_v^t] \\
&= \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t [w_{u,v} + q_v^t] \\
&= \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t w_{u,v} + \sum_{v\in\mathbb{V}} q_v^t \sum_{u:(u,v)\in\mathbb{E}} f_{u,v}^t \\
&= \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t w_{u,v} + \sum_{u\in\mathbb{V}} q_u^t r_u^t - \sum_{u\in\mathbb{V}} D(u,t) q_u^t
\end{aligned}$$

The first equality is obtained using equation (3.1) and the second equation is obtained by applying the relationship between the inflow rate to a node, split ratios and link flow rates while the last equation is obtained by applying the definition of the inflow rate. Cancelling terms on both sides and rearranging yields the desired result. □

The lemma analytically states the intuitive idea that the total price of sending traffic to meet the demand in the network, as defined by the sum of the products of the traffic demand rate and the node price for each demand node, is equal to the sum over all links of the price of sending traffic through each link. The next lemma describes how to calculate the rate of change of network cost [18].

Lemma 2.

$$\sum_{(u,v)\in\mathbb{E}} \dot{f}_{u,v}^t w_{u,v} = \sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t]$$

Proof. The time derivative of the statement of Lemma 1 is,

$$\sum_{u\in\mathbb{V}} \dot{D}(u,t)q_u^t + \sum_{u\in\mathbb{V}} D(u,t)\dot{q}_u^t = \sum_{(u,v)\in\mathbb{E}} \dot{f}_{u,v}^t w_{u,v} + \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t \dot{w}_{u,v} \tag{5.5}$$

Note that $\dot{D}(u,t) = 0$, as we assume that the demands are varying at a slower timescale than the dynamics of the algorithm. Next consider the expression,

$$\begin{aligned}
\sum_{u\in\mathbb{V}} r_u^t \dot{q}_u^t &= \sum_{u\in\mathbb{V}/t} r_u^t \sum_{v:(u,v)\in\mathbb{E}} \left\{ \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] + \alpha_{u,v}^t [\dot{w}_{u,v} + \dot{q}_v^t] \right\} \\
&= \sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] + \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t [\dot{w}_{u,v} + \dot{q}_v^t] \\
&= \sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] + \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t \dot{w}_{u,v} + \sum_{v\in\mathbb{V}} \dot{q}_v^t \sum_{u:(u,v)\in\mathbb{E}} f_{u,v}^t \\
&= \sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] + \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t \dot{w}_{u,v} + \sum_{u\in\mathbb{V}} \dot{q}_u^t r_u^t - \sum_{u\in\mathbb{V}} D(u,t)\dot{q}_u^t
\end{aligned}$$

Cancelling terms and rearranging we get,

$$\sum_{u\in\mathbb{V}} D(u,t)\dot{q}_u^t = \sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] + \sum_{(u,v)\in\mathbb{E}} f_{u,v}^t \dot{w}_{u,v}$$

which we can subsitute into Equation 5.5 to get the desired result. □

The above lemma captures the fact that the change in network cost can either be expressed in terms of the change in the link flow rates, i.e., how each link affects the network cost or in terms of the change in the split ratios at each node, i.e., how each node affects the network cost. Now we are finally in a position to prove the main result of the dissertation which is summarized in the following theorem.

Theorem 1. *In a network, at every node u, for every destination t, let the evolution of the split ratios be defined by equations* $(5.1) - (5.4)$*. Then starting from any initial conditions we have,*

Convergence: α converges to the largest invariant set in $\{\alpha | \dot{\Phi}(f) = 0\}$

Optimality: any element of this set yields an optimal solution to the MCF problem.

Proof. We will prove the result in three steps. First, we will first show that $\dot{\Phi}(f) \leq 0$, which is the key step of the whole proof. Then, we will use this result to invoke LaSalle's Invariance Principle for hybrid systems [27] to argue that α converges to the largest invariant set in $\{\alpha | \dot{\Phi}(f) = 0\}$. Lastly, we will establish that any element of this set is an optimal solution to the MCF problem.

In order to show that $\dot{\Phi}(f) \leq 0$, note that,

$$\dot{\Phi}(f) = \sum_{t \in \mathbb{V}} \sum_{(u,v) \in \mathbb{E}} \dot{f}^t_{u,v} w_{u,v} = \sum_{t \in \mathbb{V}} \dot{\Phi}^t(f)$$

where $\dot{\Phi}^t(f) = \sum_{(u,v) \in \mathbb{E}} \dot{f}^t_{u,v} w_{u,v}$ is the rate of change of the network cost as the flows to destination t change. Consequently, if we show that $\dot{\Phi}^t(f) \leq 0$ for each destination t, then we have that $\dot{\Phi}(f) \leq 0$. From Lemma 2 we know that,

$$\dot{\Phi}^t(f) = \sum_{(u,v) \in \mathbb{E}} \dot{f}^t_{u,v} w_{u,v} = \sum_{u \in \mathbb{V}} \sum_{(u,v) \in \mathbb{E}} r^t_u \dot{\alpha}^t_{u,v} [w_{u,v} + q^t_v]$$

The key idea behind the proof is to decompose the change in cost to a particular destination t, by grouping the terms from the summation derived in Lemma 2, using the branches of the shortest path tree rooted at that destination. More precisely, we define a *branch* ($\mathcal{B}$) as the set of routers on the path from a leaf router on the shortest path tree to the destination router t. Given the definition, it is easy to see that some intermediate routers will be shared among multiple branches. The change in cost contributed by these routers is divided among the different branches that pass through these routers in the following way. Each router u has a corresponding η^t_u value which appears in the denominator of the expression for the change in cost. The idea is, when grouping terms, for a particular

branch passing through an intermediate router, to only take a fraction, $1/\pi_u^{\mathcal{B}}$, of the change in cost contributed by the intermediate router, to be summed with that branch so that $\pi_u^{\mathcal{B}}\eta_u^t$ for that router u is the same as the branch cardinality of the leaf router which defines the branch. Consequently, $\pi_u^{\mathcal{B}}\eta_u^t$ will be the same for all routers u encountered in a traversal from the leaf router of the branch to the destination. Armed with this information, we further break down the above summation as follows.

$$\sum_{u\in\mathbb{V}}\sum_{(u,v)\in\mathbb{E}} r_u^t\dot{\alpha}_{u,v}^t[w_{u,v}+q_v^t] = \sum_{\forall\mathcal{B}}\sum_{u\in\mathcal{B}}\frac{1}{\pi_u^{\mathcal{B}}}\sum_{(u,v)\in\mathbb{E}} r_u^t\dot{\alpha}_{u,v}^t[w_{u,v}+q_v^t]$$

For a given branch $\mathcal{B}$, with n routers numbered $1,\ldots,n$ from the leaf router to the destination, as noted above, $1/\pi_u^{\mathcal{B}}$ is the fraction of the change in cost due to router u that it contributes to the branch summation. For ease of notation, in what follows, we will use η to represent $\pi_u^{\mathcal{B}}\eta_u^t$ for every router u that belongs to the branch $\mathcal{B}$. Then, in order to establish that, $\sum_{u\in\mathbb{V}}\sum_{(u,v)\in\mathbb{E}} r_u^t\dot{\alpha}_{u,v}^t[w_{u,v}+q_v^t]\leq 0$ it is sufficient to show that,

$$\sum_{u\in\mathcal{B}}\frac{1}{\pi_u^{\mathcal{B}}}\sum_{(u,v)\in\mathbb{E}} r_u^t\dot{\alpha}_{u,v}^t[w_{u,v}+q_v^t]\leq 0$$

for all the branches $\mathcal{B}$ of the shortest path tree which is true since,

$$
\begin{aligned}
&\sum_{u\in\mathcal{B}} \frac{1}{\pi_u^{\mathcal{B}}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] \\
=& - r_1^t \sum_{(1,v)\in\mathbb{E}} \frac{\alpha_{1,v}^t \delta}{\eta r_1^t} [w_{1,v} + q_v^t] + r_1^t \sum_{(1,v)\in\mathbb{E}} \frac{\alpha_{1,v}^t \delta}{\eta r_1^t} [w_{1,2} + q_2^t] \\
&- r_2^t \sum_{(2,v)\in\mathbb{E}} \frac{\alpha_{2,v}^t \delta}{\eta r_2^t} [w_{2,v} + q_v^t] + r_2^t \sum_{(2,v)\in\mathbb{E}} \frac{\alpha_{2,v}^t \delta}{\eta r_2^t} [w_{2,3} + q_3^t] \\
&- \cdots - r_{n-1}^t \sum_{(n-1,n)\in\mathbb{E}} \frac{\alpha_{n-1,v}^t \delta}{\eta r_{n-1}^t} [w_{n-1,v} + q_v^t] \\
&+ r_{n-1}^t \sum_{(n-1,v)\in\mathbb{E}} \frac{\alpha_{n-1,v}^t \delta}{\eta r_{n-1}^t} w_{n-1,n} \\
=& - \frac{\delta}{\eta} q_1^t + \frac{\delta}{\eta} [w_{1,2} + q_2^t] - \frac{\delta}{\eta} q_2^t + \frac{\delta}{\eta} [w_{2,3} + q_3^t] \\
&- \cdots - \frac{\delta}{\eta} q_{n-1}^t + \frac{\delta}{\eta} w_{n-1,n} \\
=& - \frac{\delta}{\eta} [q_1^t - w_{1,2} - \cdots - w_{n-1,n}] \leq 0
\end{aligned}
$$

The last inequality follows from the fact that the average price from the leaf router (node 1) to the destination (node n) which can be thought of as an average over paths from Equation (3.2) has to be no less than the price of the shortest path. Note that this relationship holds with equality only when the node price of the leaf node is the same as the price of the shortest path which means that all the traffic from every node in the branch to the destination is along shortest paths to the destination.

However, in order to ensure the chain cancellation of node prices that led to the above result, every node in $\mathcal{B}$ has to be executing the algorithm which is something that HALO requires. Another concern is that it is possible some of the routers u that belong to $\mathcal{B}$ might have $r_u^t = 0$, which seems to interfere with the cancellation of the intermediate node prices. Fortunately, this is not a problem since the algorithm will send all the incoming traffic along the selected shortest

path at nodes which did not previously have traffic to the destination. In order to see why the node price cancelation is not affected, suppose that a subset $\mathcal{B}_r$ of $\mathcal{B}$ has $r_u^t = 0$. Evaluating the branch summation we get,

$$\sum_{u \in \mathcal{B}/\mathcal{B}_r} -\frac{\delta}{\eta}(q_u^t - w_{u,u+1} - q_{u+1}^t)$$

where for $q_{u+1}^t \in \mathcal{B}_r$, we take advantage of our additional knowledge about the routing at nodes belonging to $\mathcal{B}_r$, specifically that they forward all incoming traffic through the shortest path to write $q_{u+1}^t = w_{u+1,u+2} + q_{u+2}^t$ giving us,

$$-\frac{\delta}{\eta}[q_1^t - w_{1,2} - \cdots - w_{n-1,n}] \leq 0$$

Having established that for any branch the change in cost is at most zero, we can say that the change in cost for the shortest path tree which is composed of multiple such branches is at most zero as well. The total change in cost is just the sum of the changes in cost over all destinations which will also be negative. i.e.,

$$\dot{\Phi} = \sum_t \dot{\Phi}^t(f) = \sum_{(u,v) \in \mathbb{E}} \dot{f}_{u,v}^t \Phi'(f_{u,v}) \leq 0 \tag{5.6}$$

Thus, given the control laws we have established that $\dot{\Phi}(f) \leq 0$. Now, having earlier established that our system is an example of a non-blocking, deterministic and continuous hybrid automaton, we can apply a generalization of LaSalle's Invariance Principle to hybrid automata [27], to show that the set of split ratios converges to the largest invariant set within $\{\alpha | \dot{\Phi}(f) = 0\}$.

In order to see why this leads to an optimal solution note that $\dot{\Phi}(f) = \sum_{t \in \mathbb{V}} \dot{\Phi}^t(f)$. We have already shown that each element of this summation is at most zero along the trajectory. Thus, the only way to have $\dot{\Phi}(f) = 0$ is if each

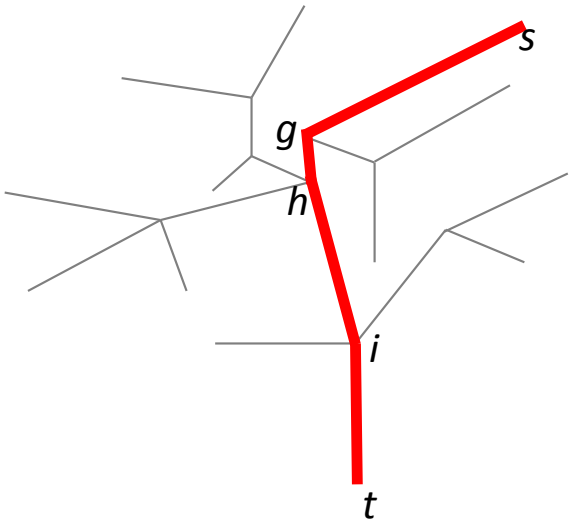


Figure 5.1: **Shortest Path Tree.** Only the links in the shortest path tree for terminal t is shown with the other links in the network not shown for ease of exposition.

$\dot{\Phi}^t(f) = 0$ which implies that the change in cost along each branch,

$$\sum_{\substack{(u,v)\in\mathbb{E} \\ \text{such that } u\in\mathcal{B}}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] = 0$$

for every t. From the preceding analysis, the change in cost along a branch $\mathcal{B}$ is zero only when all the traffic from the nodes that belong to the branch is being routed to the destination through shortest paths with respect to the link prices. Since this is a necessary and sufficient condition for optimality in MCF [1], the proof is complete. □

Next, as an illustrative example to help understand the first step of the above proof, we consider a sample shortest path tree and perform the corresponding cost change calculations explicitly. Consider the shortest path tree of Figure 5.1. The number of branches that we divide the tree into is determined by the number of leaf nodes. In this example, the shortest path tree rooted at t has 12 leaf routers and consequently we will divide the summation into 12 branches. Following the

algorithm for the calculation of η, we find, $\eta_i^t = 1,\ \eta_h^t = 3,\ \eta_g^t = 9$ and $\eta_s^t = 18$. As noted in the proof, the change in the cost function due to the routers increasing traffic along the links in the shortest path tree can be calculated using Lemma 2. In order to evaluate it, we further divide the terms in the summation and group them per branch. Recall from the proof that for the routers that are downstream to a leaf router in a branch, only a fraction of the change in the cost contributed by the downstream router is selected where the fraction is determined by the need to have the same η for all routers in the summation for a branch. The contribution to the change in the cost by the routers for the highlighted branch can be calculated as follows,

$$
\begin{aligned}
&\sum_{u\in\mathcal{B}} \frac{1}{\pi_u^{\mathcal{B}}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] \\
=& - r_s^t \sum_{(s,v)\in\mathbb{E}} \frac{\alpha_{s,v}^t \delta}{\eta_s^t r_s^t} [w_{s,v} + q_v^t] + r_s^t \sum_{(s,v)\in\mathbb{E}} \frac{\alpha_{s,v}^t \delta}{\eta_s^t r_s^t} [w_{s,g} + q_g^t] \\
&- r_g^t \sum_{(g,v)\in\mathbb{E}} \frac{\alpha_{g,v}^t \delta}{2\eta_g^t r_g^t} [w_{g,v} + q_v^t] + r_g^t \sum_{(g,v)\in\mathbb{E}} \frac{\alpha_{g,v}^t \delta}{2\eta_g^t r_g^t} [w_{g,h} + q_h^t] \\
&- r_h^t \sum_{(h,v)\in\mathbb{E}} \frac{\alpha_{h,v}^t \delta}{6\eta_h^t r_h^t} [w_{h,v} + q_v^t] + r_h^t \sum_{(h,v)\in\mathbb{E}} \frac{\alpha_{h,v}^t \delta}{6\eta_h^t r_h^t} [w_{h,i} + q_i^t] \\
&- r_i^t \sum_{(i,v)\in\mathbb{E}} \frac{\alpha_{i,v}^t \delta}{18\eta_i^t r_i^t} [w_{i,v} + q_v^t] + r_i^t \sum_{(i,v)\in\mathbb{E}} \frac{\alpha_{i,v}^t \delta}{18\eta_h^t r_i^t} [w_{i,t}] \\
=& - \frac{\delta}{\eta_s^t} [q_s^t - w_{s,g} - w_{g,h} - w_{h,i} - w_{i,t}] \leq 0
\end{aligned}
$$

CHAPTER 6

ADDITIONAL ASPECTS OF HALO

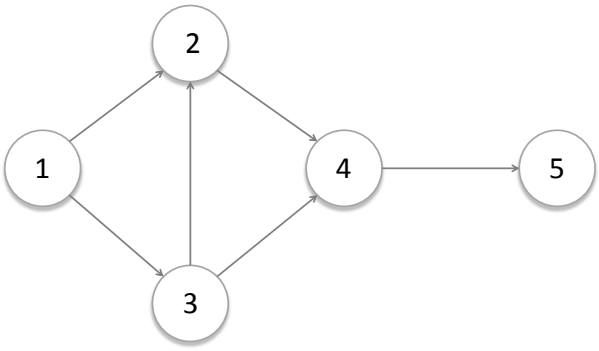
Having described HALO and proved its convergence and optimality, we now discuss some other facets of it.

6.1 Different Algorithms Can End Up With Different Split Ratios

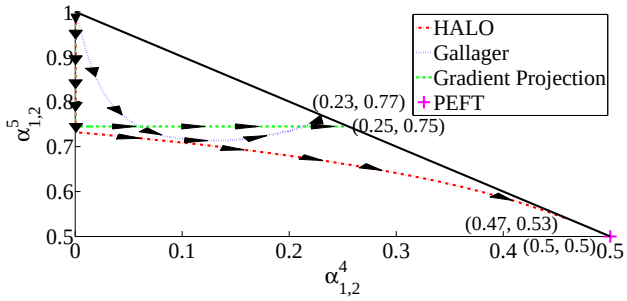
As shown in Figure 4.5, the new algorithm follows a different trajectory from Gallager's algorithm in searching for an optimal solution. But in that case, both algorithms converged to the same optimal solution. In general, because MCF problem is strictly convex in link rates ($f_{u,v}$) and only convex in flow rates ($f^t_{u,v}$), there can be multiple optimal solutions in terms of the flow rates. We demonstrate this with the following example. Figure 6.1a shows the network topology that was studied. Each link has capacity of 5 and there are two demands $D(1,4) = D(1,5) = 2$. The initial routes supplied to the different algorithms are (1-3-2-4) and (1-2-4-5), i.e., $\alpha^4_{1,2} = 0$ and $\alpha^5_{1,2} = 1$. As can be seen from Figure 6.1b, each algorithm generates a different optimal solution, all of which satisfy $\alpha^4_{1,2} + \alpha^5_{1,2} = 1$; an optimality condition which follows from the fact that at optimum, $f_{3,2} = 0$ and the resulting symmetry of the problem.

6.2 Discrete-time Implementation

Although we proved the optimality of HALO using continuous time arguments, any implementation of the protocol will need to use discrete time steps. Typically, once



(a) Network used to study algorithm trajectories with initial routes shown



(b) Trajectories taken by different algorithms to find the optimal solution

Figure 6.1: Trajectory Comparison

a continuous time argument is available, it is enough to expect that with a small enough step size, similar results should hold in discrete time as well. However, HALO exhibits hybrid dynamics where this intuition is sometimes not accurate [28]. Fortunately, our extensive numerical and experimental evaluations in Chapter 7 and Chapter 8 indicate that this is not the case and that the algorithm does in fact converge to the optimal solution even in discrete time.

6.3 Interaction with Single Path Routing

Before we delve into the analysis of how HALO interacts with a single-path routing protocol, it is necessary to clearly state the assumptions that we use and to establish the set up of interest to us. Firstly, by a single-path protocol used to make routing decisions, we mean that the router uses a set of link weights to calculate the shortest path to the destination and makes forwarding decisions based on that shortest path. Also, if the single-path protocol calculations are triggered as often as that of HALO, it is easy to construct examples where the routes in the network will oscillate and not settle down. This is because the single-path protocol moves all the traffic from one path to another instead of just a fraction. So, secondly, it is important to have a notion of time-scale separation between how often HALO is triggered and the single-path protocol is triggered. In our set up, it is assumed that the subset of routers running HALO will execute the algorithm in between slower single-path protocol calculations.

Given this set up, the two protocols can work with either the same link weights or protocol specific link weights. Since local optimization algorithms exist for calculating single-path protocol link weights and because protocol specific calculations can be triggered on the receipt of new protocol specific link weights, we assume the use of protocol specific link weights that are broadcast by each router at different time-scales. However, this assumption is more important from an implementation perspective than for the argument that follows. Another useful assumption is that each router is aware of the protocol that the other routers in the network are using.

With the time-scale separation and the assumption that every router is aware of the protocol running at every other router, for a given destination, we can see that the 'single-path' routers have a pruning affect on the network from the

perspective of the HALO routers, i.e., the outgoing links that are not used by them are effectively not a part of the network topology. Assuming that every router is aware of the protocol running at every other router, the nodes running HALO will base their calculations on this reduced network and attain the optimal routing solution for this network. Essentially, the routers implementing HALO increase the search space for finding a better routing solution and thus improve network performance.

CHAPTER 7

NUMERICAL EVALUATION AND RESULTS

In this chapter, we consider numerical evaluations of the performance of HALO from the point of view of optimality and rate of convergence to the optimal solution. We also present evidence of the adaptivity of the algorithm as the traffic changes as well as studies of the performance of HALO in asynchronous evironments and its interaction with single path routing protocols. The evaluations are primarily performed on three networks – the benchmark Abilene network (Figure 7.1), a 4×4 Mesh network and a two-level hierarchical 50 node network [3]. The 4×4 Mesh network is selected to study the affects of intermediate routing loops on the optimality of the algorithm as this topology is particularly prone to such loops while the hierarchical network is selected to mimic larger networks with high capacity backbone links and lower capacity local links. An additional test is performed on an even larger randomly generated 100 node network in order to confirm that the algorithm converges quickly for large networks (Figure 7.4). Randomly generated traffic demands are used for the mesh network and the hierarchical network while for the Abilene network uniform traffic demand is used. In order to study the algorithms' performance, in all three cases, the demand is scaled up till at least one link in the network is close to saturation at the optimal solution.

7.1 Convergence

As expected, the speed of convergence depends on the step-size. Here, the metric, network load, is defined as the ratio of the total traffic on the network to its total capacity. In general, smaller step-sizes guarantee convergence of the algorithm to the optimal solution at the expense of speed of convergence. This is demonstrated

Figure 7.1: Abilene Network

to be the case in Figure 7.3. But, as can be seen in Figure 7.3a and Figure 7.3c, larger step-sizes quickly approach the optimal solution though they can be prone to oscillations which prevent convergence to optimality. Often, it is sufficient to come to some neighborhood of the optimal solution and in such cases, exact convergence ceases to be an issue and small oscillations around the optimal solution are acceptable. In such situations, a larger step-size may be used. It is encouraging to note that in all our test cases, including for the large 100 node network (Figure 7.4) the algorithm was fairly quick, converging to a small neighborhood of the optimal solution within a few hundred iterations.

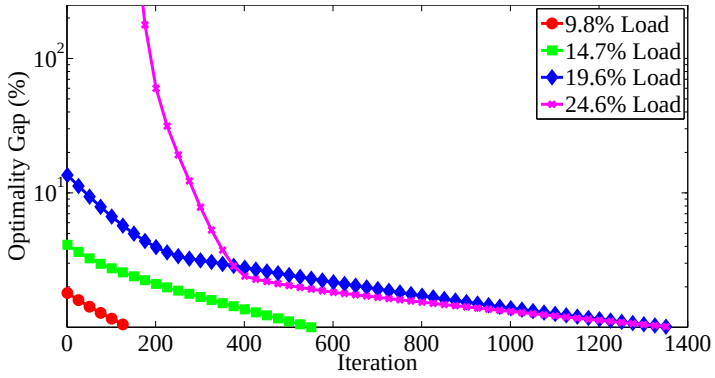
Another factor that affects the rate of convergence of the algorithm is the load on the network. The maximum network load for the Abilene network is 24.6%, mesh network is 26.1% and the hierarchical network is 5.3%. These values indicate the point at which further scaling up the demand for the given traffic pattern would exceed the capacity of at least one link in the network, even with optimal

routing. From Figure 7.2, we can see that the algorithm takes more iterations to converge to the optimal solution for more heavily loaded networks which matches intuition. As can be seen, HALO converges to the optimal solution quickly even in such limiting cases.

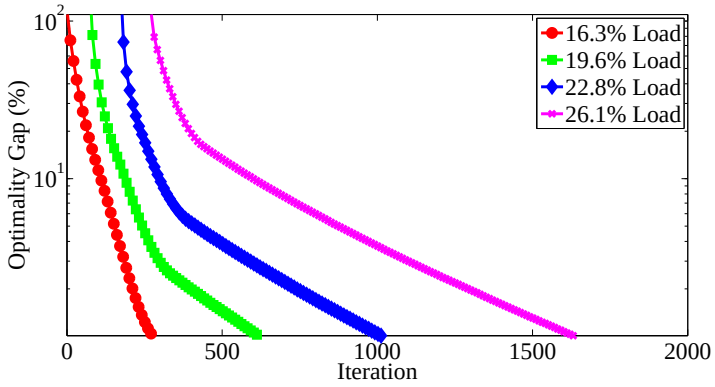
7.2 Performance

In order to verify that the algorithm does in fact achieve the optimal solution, the optimal solution was calculated for the test networks by solving the corresponding MCF problem using cvx [29] under different network load conditions. The objective value obtained by using HALO matched the optimal solution for each test case as can be seen from Figures 7.5a, 7.5b and 7.5c. Also, as expected from theory the intermediate routing loops produced while determining the optimal solution for the mesh network did not affect the optimality of the algorithm.

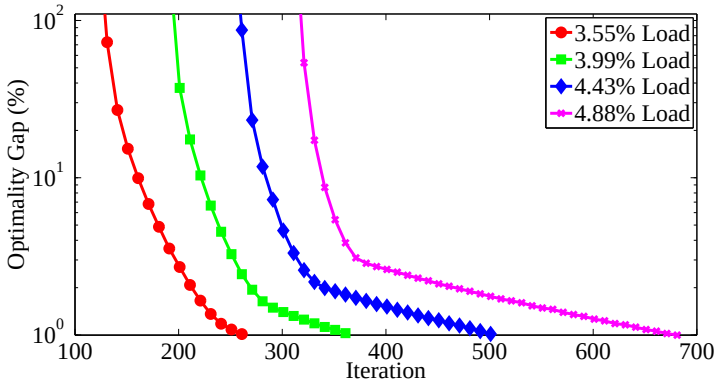
The major advantage that HALO offers is the significant performance improvement that an optimal solution offers over sub-optimal techniques like OSPF even when it is aided by locally optimal weight settings. In Figure 7.6, we compare the performance of HALO with OSPF boosted by better weight settings obtained from the algorithms of the TOTEM toolbox [30] for demand matrices that placed increasing loads on the test networks. The local search algorithm used by TOTEM minimizes a piecewise-linear approximation of our convex cost function. The power of optimality is demonstrated by the performance improvements on the order of 1000%.



(a) Abilene

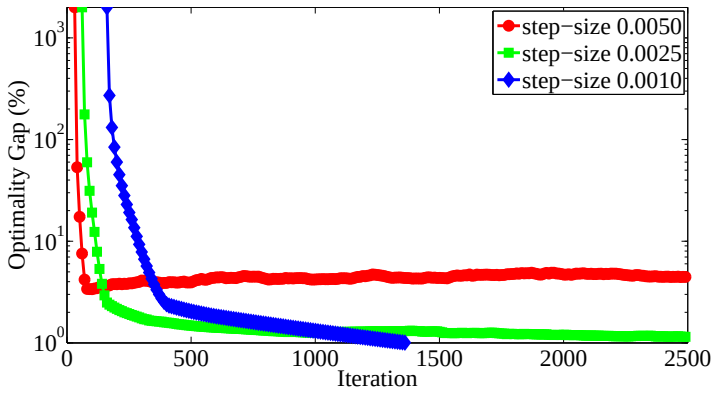


(b) 4×4 Mesh network

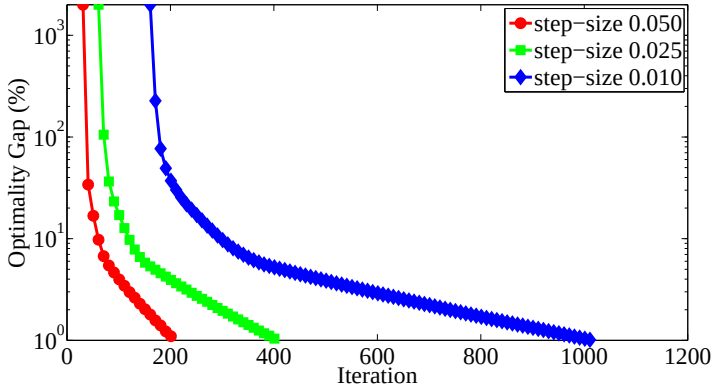


(c) Hierarchical 50 node network

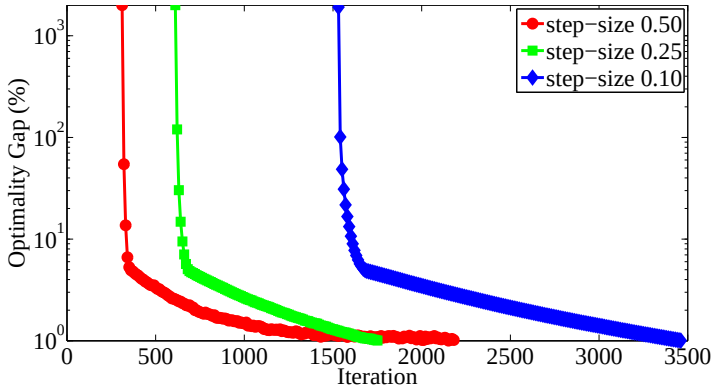
Figure 7.2: (a)Evolution of the optimality gap for the Abilene network as the number of iterations increases with different network loads (step-size = 0.001) (b) Evolution of the optimality gap for the 4×4 Mesh network as the number of iterations increases with different network loads (step-size = 0.01) (c)Evolution of the optimality gap for the Hierarchical 50 node network as the number of iterations increases with different network loads (step-size = 0.4)



(a) Abilene



(b) 4×4 Mesh network



(c) Hierarchical 50 node network

Figure 7.3: (a) Evolution of the optimality gap for the Abilene network as the number of iterations increases with varying step-sizes (network load = 24.6%) (b) Evolution of the optimality gap for the 4×4 Mesh network as the number of iterations increases with varying step-sizes (network load = 22.8%) (c) Evolution of the optimality gap for the Hierarchical 50 node network as the number of iterations increases with varying step-sizes (network load = 5.3%).

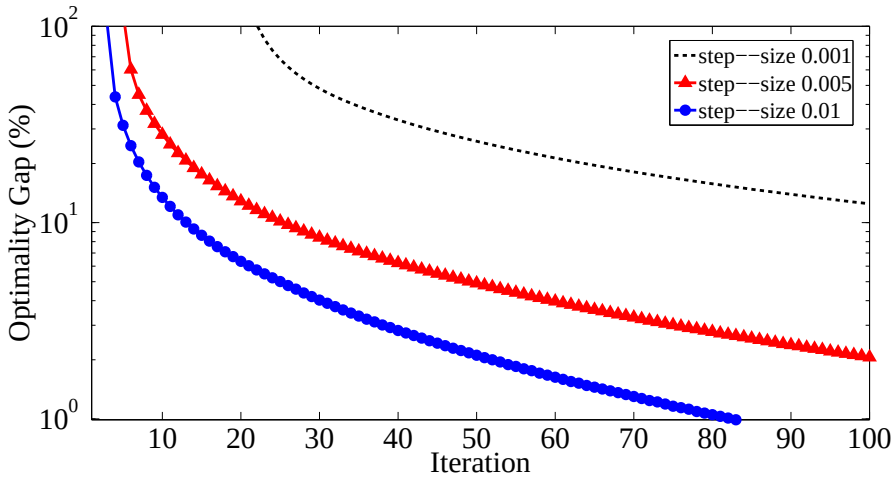
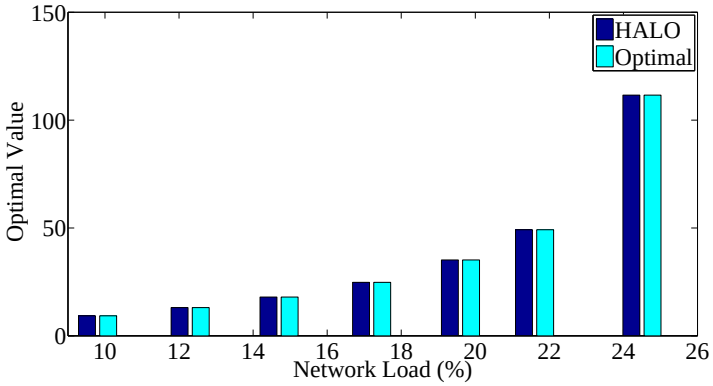


Figure 7.4: Evolution of the optimality gap for a randomly generated 100 node network with varying step-sizes

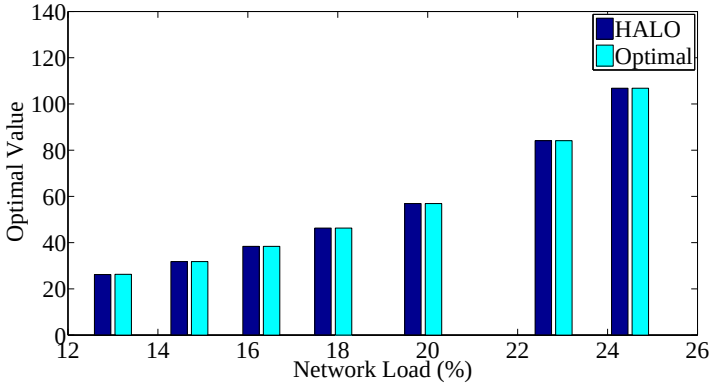
7.3 Adaptivity

Another attraction of HALO is that it dynamically adapts to changes in the traffic on the network. In Figure 7.7a, we plot the evolution of the optimality gap as the traffic matrix undergoes changes for the Abilene network under different network load conditions. In this example after around 300 iterations the network load is changed by changing 20% of the flows in the network. As can be seen, the algorithm quickly adapts and the optimality gap increases very little before beginning to converge to the new optimal solution. The traffic pattern is again changed by varying 50% of the flows in the network after 800 iterations. This time the change in the optimality gap is greater but the convergence to the new optimal value is seen to be quicker. The traffic pattern in the network was changed two more times and as can be observed from the figure in both cases the algorithm quickly converges to the new optimal solution.

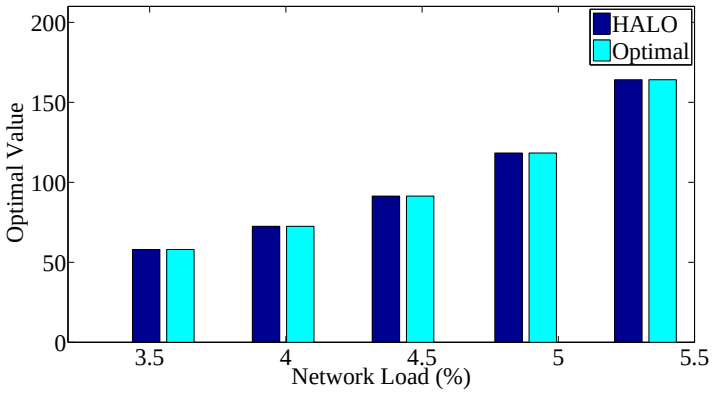
A closely related concept to the adaptivity of the algorithm is the evolution of the split ratios at individual routers. Additionally, it serves as a visualization



(a) Abilene

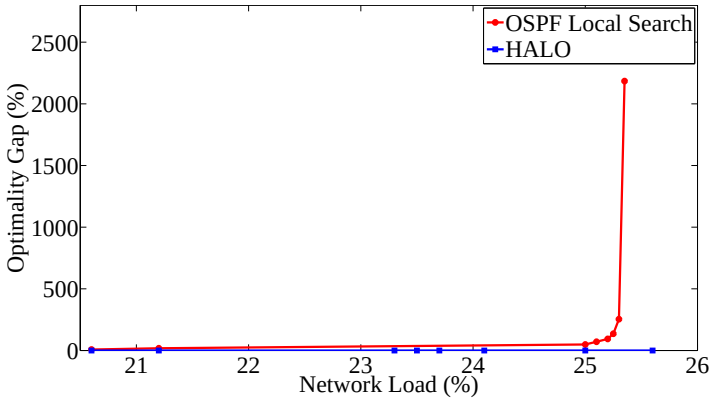


(b) 4×4 Mesh Network

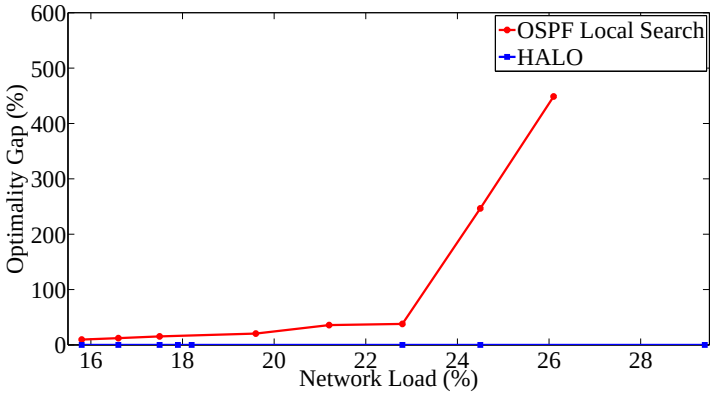


(c) Hierarchical 50 Node Network

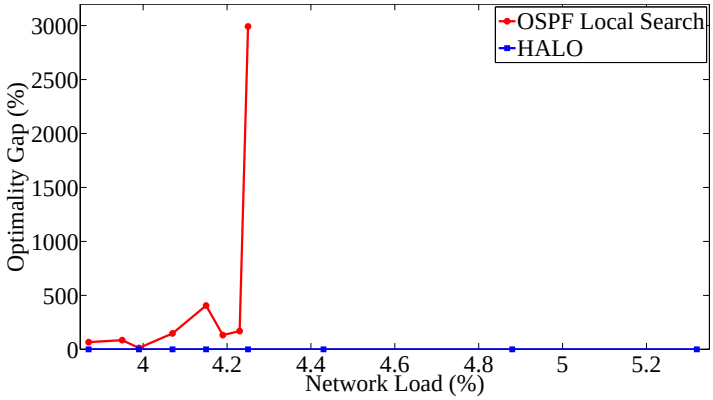
Figure 7.5: (a) Mesh Network Optimal Performance (b)Abilene Network Optimal Performance (c)Hierarchical 50 Node Network Optimal Performance – Centralized optimal value matched by HALO



(a) Abilene

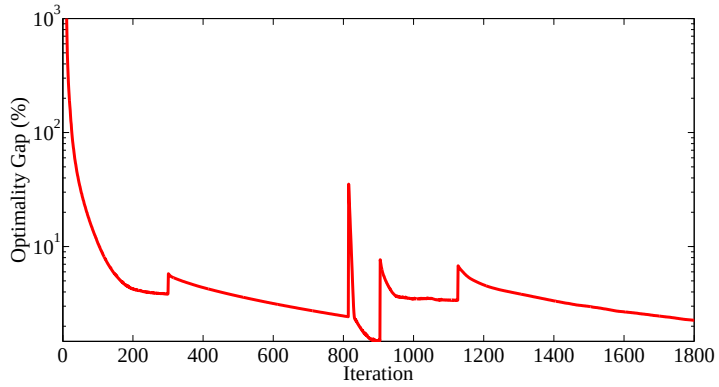


(b) 4×4 Mesh Network

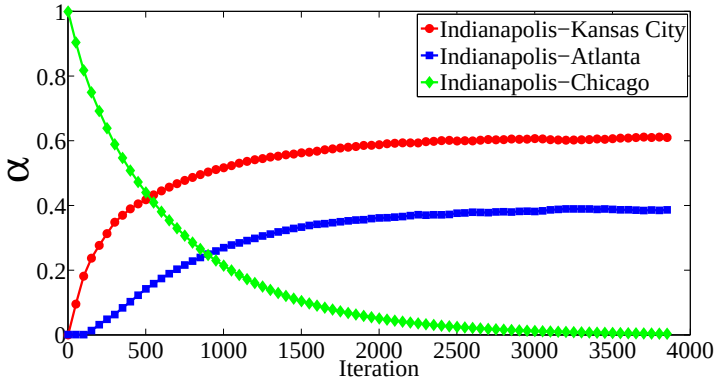


(c) Hierarchical 50 Node Network

Figure 7.6: (a) Abilene Network Algorithm Comparison (b) Mesh Network Algorithm Comparison (c)Hierarchical 50 Node Network Algorithm Comparison – Relative performance of different algorithms for different network loads



(a) Evolution of the optimality gap for the Abilene Network as the number of iterations increases with varying demand matrices



(b) Evolution of the split ratios to Chicago, Kansas City and Atlanta for traffic destined to LA at the Indianapolis node on the Abilene Network

Figure 7.7: Adaptivity of the algorithm

of HALO in action. We pick the Indianapolis node for the Abilene network and plot the evolution of the split ratios to Los Angeles in Figure 7.7b. The initial sub-optimal allocation of split ratios is quickly corrected as HALO reduces traffic sent to Chicago and increases traffic sent to Kansas City and Atlanta.

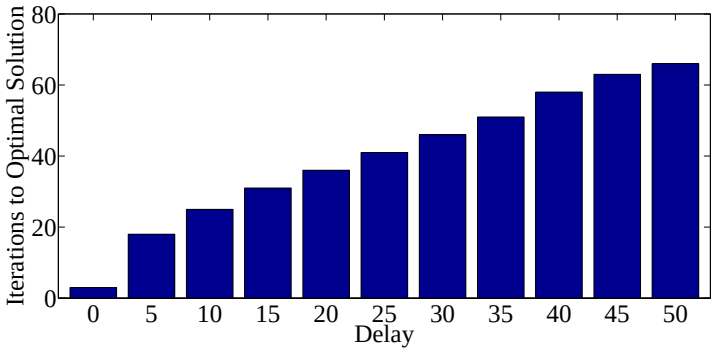
7.4 Asynchronous Implementation

In dynamic network environments, random delays can affect the time it takes for link-state information to reach every node in the network as required by the al-

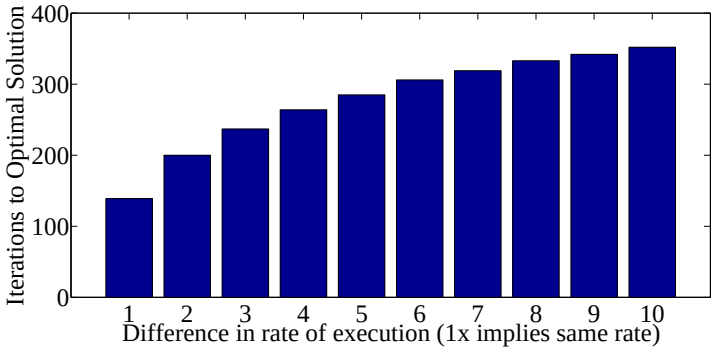
gorithm. Note that without synchronized link-state updates, facets of HALO like calculating the shortest path tree and η are affected. There are two ways to approach this problem. The first is to allow enough time between successive iterations of the algorithm so that every node has access to the most up-to-date link-state information. The second is to let the nodes execute HALO despite asynchronous link-state updates. It is also possible for asynchronous behavior to arise despite synchronized link-state updates due to some subset of the nodes executing the algorithm faster than the other nodes. We separately tested the performance of HALO, for asynchronous link-state updates and asynchronous executions, using uniform traffic on the Abilene network. The results of our evaluation, studying both type of asynchronous behavior, are presented in Figure 7.8.

In both cases, in order to simulate asynchronous behavior, the nodes in the network were numbered and divided into two groups. For asynchronous link-state updates, at every iteration, the even numbered nodes received link-states without any delay while the odd numbered nodes received link-states from the even numbered nodes after a fixed delay. Consequently, at each execution of the algorithm, the two sets of nodes had different views of the network link-states. The fixed delay was then varied to generate the results reported in Figure 7.8a. For asynchronous execution of HALO, the odd numbered nodes were forced to execute the algorithm slower than the even numbered nodes. The difference in the rate of execution was varied in order to obtain the results reported in Figure 7.8b. Note that different step-sizes had to be used to prevent oscillations in the two cases.

As can be seen, HALO still converged to within 1% of the optimal solution which was used as a stopping criterion for our evaluation. Additionally, it is interesting to note that there is a steady increase in the number of iterations required



(a) Iterations required to converge increase with increasing delay (step-size = 0.1).



(b) Iterations required to converge increase with increasing difference in rate of execution (step-size = 0.001).

Figure 7.8: Results from asynchronous implementation of HALO.

by HALO as the delay in propagating the link-states or the difference in the rate of executing HALO increases. While the results of this particular experiment are promising, more research is required to establish whether the observations noted above are a more general property of HALO.

7.5 Coexistence with Single Path Protocol

We also used numerical evaluations to confirm the predictions of Chapter 6 with respect to HALO being implemented in conjunction with a single-path routing protocol. The same setup as described in Chapter 6 is studied using a randomly

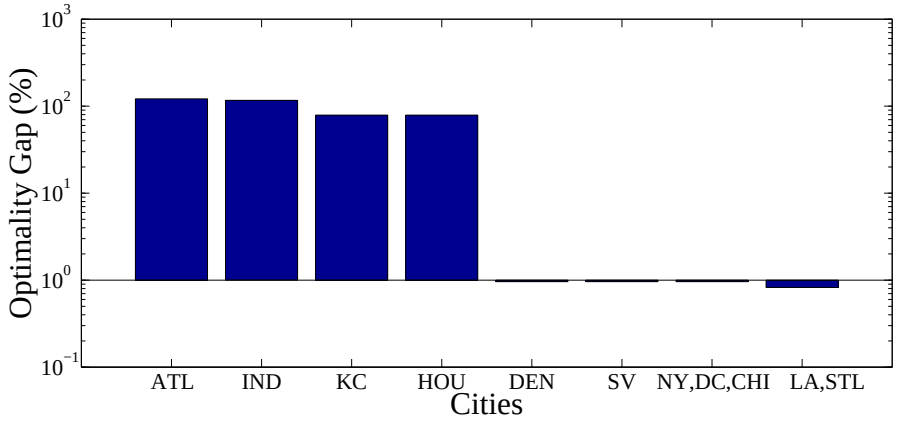


Figure 7.9: Results of partially implementing HALO on the Abilene network. Increasing the number of nodes running HALO improves performance. The horizontal axis represents more cities being added to the set implementing HALO.

generated traffic pattern on the Abilene network. The nodes were first ranked according to their degree and then added in that order to the set implementing the HALO protocol to obtain the results seen in Figure 7.9. The degree two nodes on the East and the West Coasts were added to the set together since by that point most of the performance gain had already been achieved. As can be seen, for each set implementing HALO, the algorithm converges to the optimal solution for the case where the remaining nodes are constrained to have only one out-going link per destination. As expected, as the number of cities/nodes that run the algorithm increase, the performance of the network improves. But it is also interesting how adding the first few higher degree nodes to the set implementing HALO results in the bulk of the performance improvement.

CHAPTER 8

TESTBED EXPERIMENTAL RESULTS

The promising theoretical and numerical results led us to built a hardware testbed to further confirm the theoretical and numerical predictions about HALO. Our experimental setup was a simple network of four Dell PCs connected together in the topology shown in Figure 8.1. Each computer was connected to a NetFPGA 1G platform which was programmed to act as a router. Two experiments were performed using the testbed to evaluate HALO. The first was designed to check whether the algorithm was indeed optimal, while the second was designed to see how well the algorithm adapted to changes in the network traffic. Before we describe the experimental results, we give a brief overview of how we implemented the protocol on the testbed.

8.1 Implementing HALO on NetFPGA

The starting point of our implementation of HALO on NetFPGA was the Reference Router that was designed to make the board behave as a single-path router running OSPF using hop count as the distance metric. We used the same firmware as used for the Reference Router but then the primary challenge that had to be overcome was implementing multipath forwarding using dynamically changing split ratios on the NetFPGA platform. In order to quickly achieve multipath functionality, we decided to transfer packet forwarding decisions from the firmware to higher level software which we could easily modify via SCONE (Software Component of NetFPGA).

Multipath forwarding requires split ratios to each destination and so, first we

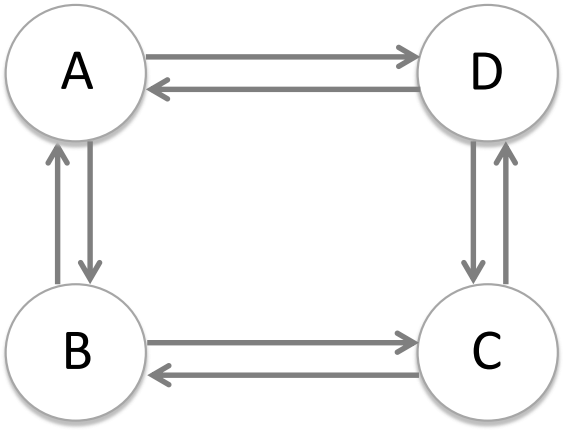


Figure 8.1: Topology of the hardware testbed. Port 1 of B leads to link (B, C) and Port 2 of B leads to link (B, A).

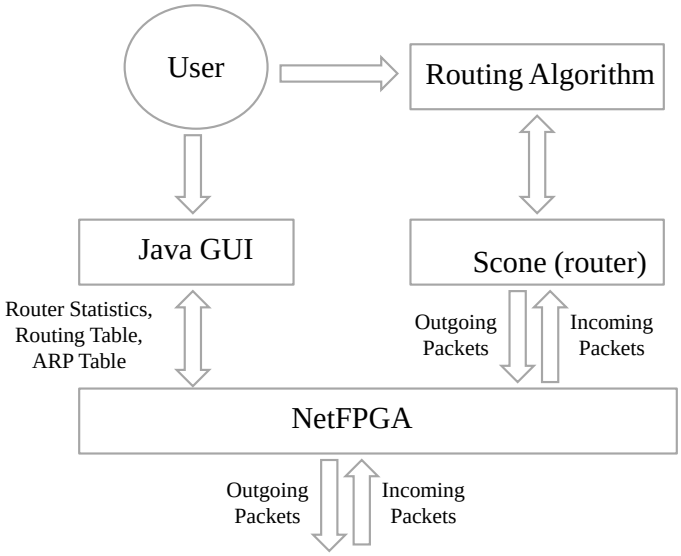


Figure 8.2: System diagram.

Destination	Mask	Port 0	Port 1	Port 2	Port 3
192.168.101.1	255.255.255.0	1	0	0	0
192.168.102.1	255.255.255.0	0	0.75	0.25	0
192.168.103.1	255.255.255.0	0	0.45	0.35	0.2

Figure 8.3: Example of a split ratio table from our experimental implementation.

created a new table at each router to store the split ratios for each destination (Figure 8.3). In general, the existing forwarding table at each router could have been modified to have the requisite number of additional columns. We chose to create a new table to minimize the exisiting code that we needed to modify. The table was designed to have as many columns as the number of outgoing links at each router and as many rows as the forwarding table already present on the router that was used for OSPF. Then, packet forwarding decisions were made using the fractions in the split ratio table in conjunction with a random number generator to determine the next hop to each destination.

Having implemented multipath, the next major challenge was modifying the link-state update packets to be broadcast frequently enough to ensure quick con-

vergence of the algorithm as well as to modify their payload to transmit the link rates. For our experiments, we set the link-states to broadcast every 250 milliseconds. To keep calculations simple, in our experiments, we used $\sum_{u,v\in\mathbb{E}} f_{u,v}^2$ as the network cost function, which gave us $2f_{u,v}$ as the price of each link. We also had to modify Dijkstra's algorithm to run with the new link weights instead of hop-count as it was doing in the Reference Router implementation in SCONE.

Additionally, note that the HALO protocol requires the calculation of η_u^t at each router u for each destination t in order to be able to calculate the split ratios that are entered into the split ratio table. The calculation of η_u^t requires knowledge of the shortest path tree that is output by Dijkstra's algorithm. For single-path routing only the next hop to a destination was required from running Dijkstra. However, for HALO, we require the entire shortest path tree and so tree data structures were used to store the shortest path trees for each destination at each iteration at each router. The value of η_u^t could then be easily calculated.

The last component required for the split ratio calculations performed by HALO was r_u^t, the incoming rate to each destination t at each router u. This quantity was calculated by implementing a per destination counter that was updated based on the destination information determined from each packet's header. Since, each packet's header needs to be examined to determine where to forward it to anyway, updating this additional entity did not add much overhead.

8.2 Evaluating Optimality of HALO

For the first experiment, we send video traffic using VLC Media Player as a video server from Computer B to Computer C. From the KKT conditions of the MCF

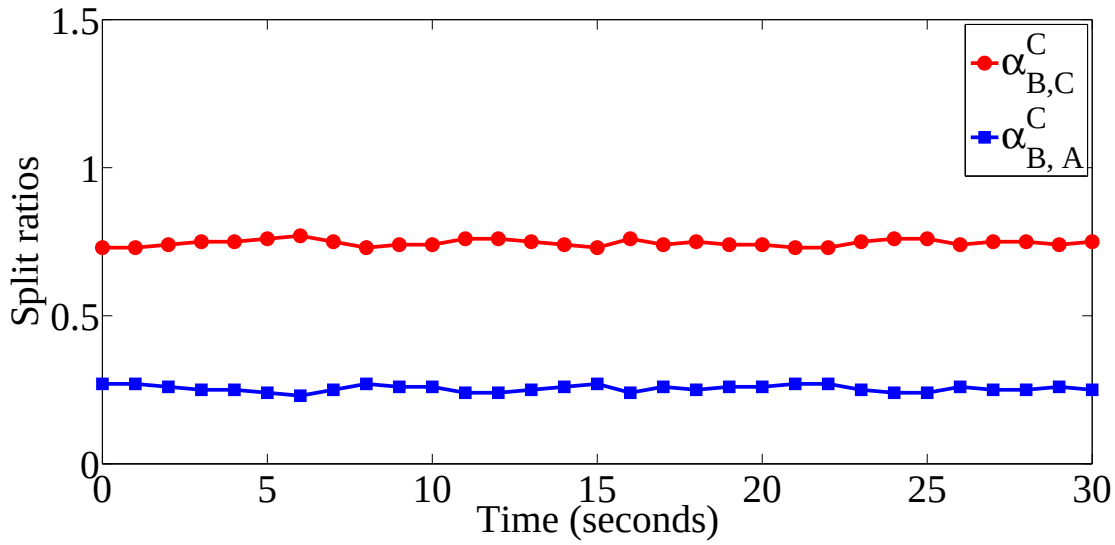


Figure 8.4: Three times as much traffic flows to Port 1 as Port 2 as expected.

problem, for the given cost function, it is easy to see that the values of the split ratios at optimality should be $\alpha^C_{B,A} = 0.25$ and $\alpha^C_{B,C} = 0.75$. The evolution of the split ratios from our experiment as captured using SCONE, which comes with the NetFPGA platform, is presented in Figure 8.4. As expected from theory, about 25% of the traffic is sent along the longer path through Port 2 while the rest is sent along the shorter path via Port 1.

8.3 Evaluating Adaptivity of HALO

For the second experiment, we used the same setup as the first experiment except that we introduced a new flow that clogged link (A, D) for about 15 seconds using the JPerf tool. The evolution of the split ratios from B to C when the heavy flow between A and D came online and then stopped is presented in Figure 8.5. Once again, theoretical predictions were verified as initially $\alpha^C_{B,C}$ increases to 1 before dropping back down to 0.75 once the large flow stops. The extra traffic that can

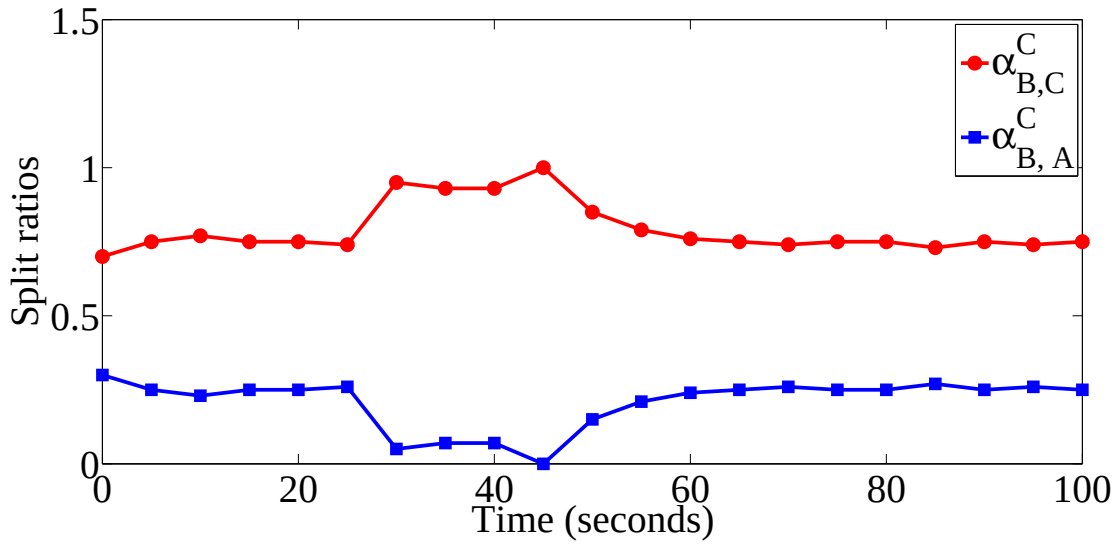


Figure 8.5: Traffic initially increases on Port 1 and then decreases as the flow from A to D starts and stops.

be seen while the flow from A to D is in progress is because some of the traffic is routed via $(A, B) \rightarrow (B, C) \rightarrow (C, D)$ as expected from the behavior of the protocol. However, most of the traffic from B to C is clearly routed via Port 1.

CHAPTER 9

CONCLUSION

We set out to develop a link-state, hop-by-hop routing algorithm that optimally solves the traffic engineering problem for intra-domain routing on the internet. Initially, a review of the rich literature on this topic pointed us towards the challenges in developing such a protocol. Fortunately, starting with simple examples we identified a new way of thinking about the problem i.e., at each node to shift traffic to the shortest path from it to the destination. By adapting this idea to different network conditions using examples of increasing complexity, we arrived at HALO, a hop-by-hop dynamic update rule that could be computed from link-states. Furthermore, the link weights for HALO can be computed directly from link rates without the traffic matrix as an explicit input. As a result, the algorithm can automatically adapt to traffic demand changes by adjusting router split ratios.

Using a creative breakdown of the complex expression for the change in the network cost, we established that HALO continuously decreases the cost of the network. Then using hybrid systems theory we were able to establish the optimality of the protocol. In order to better understand HALO vis-à-vis existing protocols, we constructed and studied examples where different protocols followed different trajectories to reach different optimal solutions. We also discussed how even a subset of routers running HALO would improve performance in a network where the other routers used single path routing. From an implementation perspective this is an attractive quality as it allows for an incremental rollout.

Numerical and experimental evaluations were used to verify theoretical predictions about HALO regarding optimality and adaptivity. The evaluations were especially useful in studying aspects such as the rate of convergence to optimality

of the protocol as well as its robustness to asynchrony in the network.

In terms of future directions, there are still important areas to be explored. For instance, the convergence rate of the algorithm needs to be analyzed. Another interesting direction involves developing the theory behind the performance of algorithm in the absence of synchronous link-state updates and executions. Lastly, we feel that HALO and the ideas behind its design have applications to network problems in other fields as well and that this is another promising area to explore. As a concrete example, we leave the reader with a potential application to road transportation networks.

The fundamental difference between a road network and a communication network is that at a junction on a road network, there is no router to tell cars which way to go. Suppose instead that we consider a system where some cars have 'smart' computers that tell them which way to turn at a junction based on road congestion information fed into our algorithm. The interaction with drivers not implementing this system is now an important consideration in deciding whether this is a good idea. An implicit assumption that we make here is that demand is constant for some period of time – a reasonable assumption during rush hour in big cities.

The preceding arguments for HALO seem to no longer work as 'routers/junctions' do not make forwarding decisions anymore. Instead, the decisions are made by the 'packets/cars'. But with one more assumption we can show that our algorithm will improve the overall network delay. Suppose that the cars which do not use the new system choose to follow the shortest path from source to destination. Note that this is a reasonable assumption to make about drivers in general and that this is equivalent to reducing the 'capacity' of each road by some fixed quantity. Following our earlier arguments, we can once again

show that the optimal routing solution can be found for this 'reduced capacity' network by some subset of the cars using the new algorithm. Of course, in addition to overcoming other technical and infrastructural challenges, implementing such a system would require some type of 'smart meter' that can measure real time traffic density on the roads and communicate this information to the computers on the cars that advice the driver. All the same, the example demonstrates intriguing potential for the application of HALO to networks in general.

APPENDIX A

OBSERVATIONS ON PEFT

A.1 Why PEFT is not optimal

In order to understand why PEFT is not optimal as claimed in [9], it is necessary to follow the analysis of the algorithm carefully. Given the necessary capacities that the different links need to have for the optimal routes calculated by the MCF problem, the key to calculating the per commodity split ratios at the nodes lies in using these results (the link utilizations $f^*_{u,v}$) to solve the network entropy maximization (NEM) problem which we now define.

$$\max_{x^i_{s,t}} \sum_{s,t} \left(D(s,t) \sum_{i \in P_{s,t}} z(x^i_{s,t}) \right)$$

$$s.t. \sum_{s,t,i} D(s,t) K^{(u,v)}_{P^i_{s,t}} x^i_{s,t} \leq \tilde{c}_{u,v} \quad \forall (u,v)$$

$$\sum_i x^i_{s,t} = 1 \ \forall s,t$$

$$x^i_{s,t} \geq 0$$

where $P_{s,t}$ represents the set of paths from s to t, $K^{(u,v)}_{P^i_{s,t}}$ represents the number of times that a link (u,v) is used by a path $P^i_{s,t}$, $x^i_{s,t}$ is the split ratio along path $P^i_{s,t} \in P_{s,t}$, $z(x) = -x \log x$ and $\tilde{c}_{u,v} = f^*_{u,v}$. Note that because the domain of the log function prohibits solutions that are not positive we can drop the constraints that explicitly require the split ratios to be positive.

The NEM problem is a concave optimization problem with a non-empty feasible set as we know the solution to the multi-commodity flow problem satisfies it. But note that we cannot solve the NEM problem directly due to the infinite

number of variables represented by all possible path rates. Instead, we consider the formulation of its dual. The Lagrangian corresponding to the NEM problem can be written as,

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = & \sum_{s,t} \left(D(s,t) \sum_{P^i_{s,t} \in P_{s,t}} z(x^i_{s,t}) \right) \\ & - \sum_{(u,v)\in\mathbb{E}} \lambda_{u,v} \left(\sum_{s,t,i} D(s,t) K^{(u,v)}_{P^i_{s,t}} x^i_{s,t} - \tilde{c}_{u,v} \right) \\ & - \sum_{s,t} \mu_{s,t} \left(\sum_i x^i_{s,t} - 1 \right) \end{aligned}$$

The dual problem can then be formulated as

$$\min_{\boldsymbol{\lambda}\geq 0} \max_{x^i_{s,t}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{A.1}$$

At this stage, in the original paper and its follow up [31] we can see that the dual variables $\boldsymbol{\lambda}$ are calculated via gradient descent in a distributed manner.

$$\begin{aligned} \lambda_{u,v}(q+1) &= \left[\lambda_{u,v}(q) - \alpha(q) \left(\tilde{c}_{u,v} - \sum_{s,t,i} D(s,t) x^i_{s,t} \right) \right]^+ \\ &= [\lambda_{u,v}(q) - \alpha(q)(\tilde{c}_{u,v} - f_{u,v}(q))]^+ \ \forall (u,v) \in \mathbb{E} \end{aligned}$$

Next we write the KKT conditions for the NEM problem and show that as long as we know $\boldsymbol{\lambda}^*$ we can calculate $x^{i*}_{s,t}$. From the stationarity condtions we find,

$$z'(x^{i*}_{s,t}) - \sum_{(u,v)\in P^i_{s,t}} K^{(u,v)}_{P^i_{s,t}} \lambda^*_{u,v} - \frac{\mu^*_{s,t}}{D(s,t)} = 0 \tag{A.2}$$

Note that dual feasibility is maintained as $\boldsymbol{\lambda}^* \geq 0$ from the gradient descent procedure. Then paying attention to primal feasibility and complementary slackness we can write,

$$x_{s,t}^{i*} = e^{-\left(\sum_{(u,v)\in P_{s,t}^i} K_{P_{s,t}^i}^{(u,v)} \lambda_{u,v}^* + \frac{\mu_{s,t}^*}{D(s,t)} + 1\right)} \tag{A.3}$$

which implies that

$$\frac{x_{s,t}^{i*}}{x_{s,t}^{j*}} = \frac{e^{-\left(\sum_{(u,v)\in P_{s,t}^i} K_{P_{s,t}^i}^{(u,v)} \lambda_{u,v}^*\right)}}{e^{-\left(\sum_{(u,v)\in P_{s,t}^j} K_{P_{s,t}^j}^{(u,v)} \lambda_{u,v}^*\right)}} \tag{A.4}$$

which leads us to conclude that we should split along the paths in $P_{s,t}$ exponentially according to their path length as defined by $\boldsymbol{\lambda}$. Using the above observation it is possible to compute that traffic splits from the expression

$$f_s^t - \sum_{x:(x,s)\in\mathbb{E}} f_x^t \left(\frac{\Gamma_{PX}(h_{x,s}^t)}{\sum_{(x,j)\in\mathbb{E}} \Gamma_{PX}(h_{x,j}^t)}\right) = D(s,t) \tag{A.5}$$

where $\Gamma_{PX}(h_{u,v}^t) = \Upsilon_v^t e^{-h_{u,v}^t}$, $h_{u,v}^t = d_v^t + \lambda_{u,v} - d_u^t$, d_u^t is the shortest distance from any node u to node t using link weights $\boldsymbol{\lambda}$ and

$$\Upsilon_u^t = \sum_{i\in P_{u,t}} e^{-(p_{u,t}^i - d_u^t)} \tag{A.6}$$

The idea behind the algorithm is to use the $\boldsymbol{\lambda}$ as link weights, to determine per commodity split ratios at the nodes based on this $\boldsymbol{\lambda}$ and to use the newly calculated split ratios to determine new link flow rates which can then be used to update $\boldsymbol{\lambda}$. It is shown that this procedure converges to the optimal split ratios and $\boldsymbol{\lambda}$.

We are now in a position to understand the potential issues with the PEFT algorithm. The more serious issue, which we will discuss in the next subsection, is that the algorithm can fail to achieve optimal traffic for most network topologies. Fortunately, a quick fix exists which will let the algorithm come arbitrarily

close to achieving the optimal traffic distribution. A less critical issue, given the centralized calculation of necessary capacities, is that the algorithm cannot always be implemented iteratively in a distributed manner. Unless the $\boldsymbol{\lambda}$ are not above a certain threshold, a centralized calculation of the optimal $\boldsymbol{\lambda}$ becomes necessary as we will show in the following section.

The subtle reason why PEFT can lead to sub-optimal solutions is that the NEM objective function is not differentiable at the optimum when one of the optimal paths has zero rate. Consider the examples in Figure A.1 and Figure A.2 to see simple examples where PEFT fails to find the optimal solution. In both cases PEFT fails to find the optimal traffic distribution as it tries to send traffic along all possible paths for each source-destination pair.

In the case of the DAG, it is easy to see why the solution generated by PEFT will not be optimal. We will explore the example with the cycle in some more detail. Here PEFT will route some traffic from A to T through link C $\rightarrow$ B as it tries to send non-zero rate through all paths. The error in both cases occurs because in deriving the optimal path split in equation (A.3), there is an implicit assumption that all paths will have non-zero rate along them. This results from the fact that $z'(x^{i*}_{s,t})$ in equation (A.2) is ∞ at $x^{i*}_{s,t} = 0$.

As discussed earlier, there is an easy fix that allows us to circumvent this problem. Note that an ϵ-perturbation to the capacity constraint of the NEM problem will take care of the above problem while providing a $(1 + \epsilon)$-optimal solution in terms of the link rates to the original optimization problem. This is easy to see since the NEM objective tries to split traffic along all possible paths. Consequently, an ϵ-perturbation to the capacity constraint allows sub-optimal paths to have an $O(\epsilon)$ non-zero rate along them which will disturb the optimal link rates by only

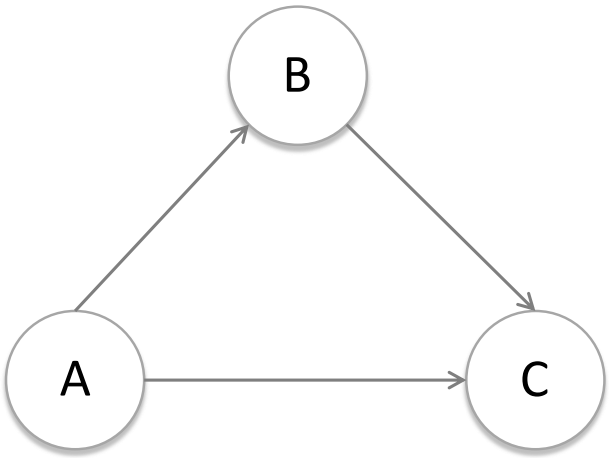


Figure A.1: Counter example to NEM in a DAG. Suppose the link capacities are $\boldsymbol{C}$. Demands D(A, B) = D(B, C) = D(A, C) = 1. The optimal solution is to send 1 unit of traffic on (A, B), (B, C) and (A, C). However, the NEM result will try to send traffic along both (A, C) and (A, B) $\rightarrow$ (B, C).

an $O(\epsilon)$ factor for ϵ small.

A.2 Why PEFT requires centralized weight calculations

Another problem with the suggested implementation of PEFT lies in the distributed calculation of link weights followed by split ratio calculations which are then used to further refine the link weights. The crux of this argument is based on the assumption that the summation in equation (A.6) converges to a finite value which is not always true as demonstrated by the example outlined in Figure A.3.

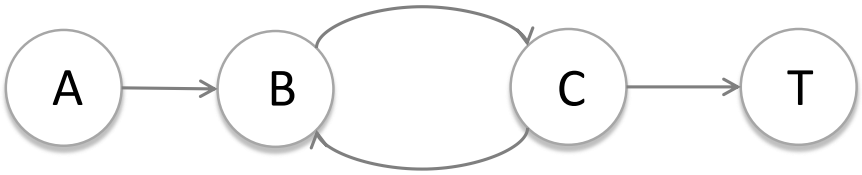


Figure A.2: Counter example to NEM with cycles. Demands D(A, T) = D(C, B) = 1. The optimal solution for this network regardless of the traffic engineering objective is for the source-destination pair of (A, T) to use the path A → B → C → T and (C, B) to use C → B.

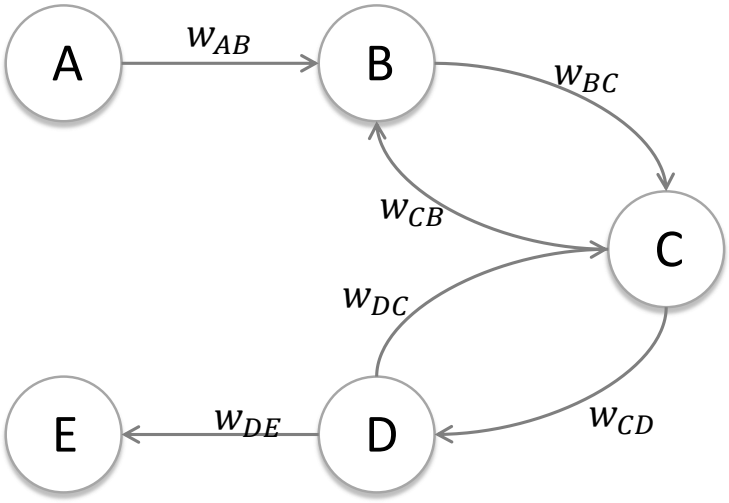


Figure A.3: Example to illustrate need for centralized weight calculations in PEFT.

We can calculate Υ_A^E for this example as follows,

$$\begin{aligned}
\Upsilon_A^E &= \sum_{i \in P_{A,E}} e^{-(p^i_{A,E} - d)} \\
&= e^d \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} e^{-\omega(m+n)} \binom{m+n}{n} \\
&= e^d \sum_{u=0}^{\infty} \sum_{n=0}^{u} e^{-\omega u} \binom{u}{n} \\
&= e^d \sum_{u=0}^{\infty} (2e^{-\omega})^u
\end{aligned}$$

It is easy to see that the above expression diverges when $\omega < \log 2$ indicating that the assumption on the basis of which the algorithm distributes calculations is not always true.

However, as noted earlier, given that the already centralized calculation of the optimal link flow rates is necessary for the algorithm, this is not a difficult issue to circumvent. The optimal link weights can be calculated centrally by computing the optimal solution to the dual of the compact NEM problem [31], and the split ratios can then be computed in a distributed manner at the nodes using this information in a single step. The issue of the convergence of the summation expression in equation (A.6) does not arise at optimum as can be seen from the analysis of the optimality conditions of the compact formulation of NEM.

Though our study of the PEFT algorithm shows how it has overcome the issue of the large data transfer required to implement a centralized solution, the protocol is still not adaptive. On the internet, an inability to adapt dynamically to the continuously varying traffic demand is a serious drawback as an optimal traffic distribution calculated for a particular traffic demand might not be close to optimal for different traffic conditions. This consideration necessitates the adaptive routing algorithm like the one developed in this dissertation.

BIBLIOGRAPHY

[1] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall, 1992.

[2] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, no. 2, pp. 97–133, 1973.

[3] B. Fortz and M. Thorup, "Increasing internet capacity using local search," *Comput. Optim. Appl.*, vol. 29, no. 1, pp. 13–48, Oct 2004.

[4] R. Gallager, "A minimum delay routing algorithm using distributed computation," *Communications, IEEE Transactions on*, vol. 25, no. 1, pp. 73 – 85, Jan 1977.

[5] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach, 5/E*. New York, NY, USA: Addison-Wesley, 2010.

[6] D. Bertsekas and E. Gafni, "Projected Newton methods and optimization of multicommodity flows," *Automatic Control, IEEE Transactions on*, vol. 28, no. 12, pp. 1090 – 1096, Dec 1983.

[7] D. Awduche, "MPLS and traffic engineering in IP networks," *Communications Magazine, IEEE*, vol. 37, no. 12, pp. 42 –47, Dec 1999.

[8] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001, pp. 1300–1309 vol.3.

[9] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1717 –1730, Dec 2011.

[10] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 2, pp. 234 – 247, Apr 2005.

[11] S. Srivastava, G. Agrawal, M. Pioro, and D. Medhi, "Determining link weight system under various objectives for OSPF networks using a lagrangian relaxation-based approach," *IEEE Trans. on Netw. and Serv. Manag.*, vol. 2, no. 1, pp. 9–18, Nov 2005.

[12] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," in *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '03. New York, NY, USA: ACM, 2003, pp. 206–217. [Online]. Available: http://doi.acm.org/10.1145/781027.781053

[13] D. Applegate and E. Cohen, "Making Routing Robust to Changing Traffic Demands: Algorithms and Evaluation," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 6, pp. 1193–1206, 2006.

[14] M. Kodialam, T. V. Lakshman, J. Orlin, and S. Sengupta, "Oblivious Routing of Highly Variable Traffic in Service Overlays and IP Backbones," *Networking, IEEE/ACM Transactions on*, vol. 17, no. 2, pp. 459–472, 2009.

[15] T. Stern, "A class of decentralized routing algorithms using relaxation," *Communications, IEEE Transactions on*, vol. 25, no. 10, pp. 1092 – 1102, Oct 1977.

[16] C. E. Agnew, "On quadratic adaptive routing algorithms," *Commun. ACM*, vol. 19, no. 1, pp. 18–22, Jan 1976.

[17] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. 1439 –1451, Aug 2006.

[18] F. Paganini and E. Mallada, "A unified approach to congestion control and node-based multipath routing," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1413–1426, Oct 2009.

[19] Y. Xi and E. Yeh, "Node-based optimal power control, routing, and congestion control in wireless networks," *Information Theory, IEEE Transactions on*, vol. 54, no. 9, pp. 4081–4106, Sep 2008.

[20] P. Mahey, A. Ouorou, L. J. LeBlanc, and J. Chifflet, "A new proximal decomposition algorithm for routing in telecommunication networks," *Networks*, vol. 31, no. 4, pp. 227–238, 1998.

[21] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *J. ACM*, vol. 37, no. 2, pp. 318–334, Apr 1990.

[22] T. Leighton, C. Stein, F. Makedon, E. Tardos, S. Plotkin, and S. Tragoudas,

"Fast approximation algorithms for multicommodity flow problems," in *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, ser. STOC '91, 1991, pp. 101–111.

[23] S. A. Plotkin, D. B. Shmoys, and E. Tardos, "Fast approximation algorithms for fractional packing and covering problems," *Math. Oper. Res.*, vol. 20, no. 2, pp. 257–301, Apr 1995.

[24] B. Awerbuch, R. Khandekar, and S. Rao, "Distributed algorithms for multicommodity flow problems via approximate steepest descent framework," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[25] M. Wang, C. W. Tan, W. Xu, and A. Tang, "Cost of not splitting in routing: characterization and estimation," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1849–1859, Dec 2011.

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[27] J. Lygeros, K. Johansson, S. Simic, J. Zhang, and S. Sastry, "Dynamical properties of hybrid automata," *Automatic Control, IEEE Transactions on*, vol. 48, no. 1, pp. 2–17, Jan 2003.

[28] ——, "Continuity and invariance in hybrid automata," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 1, 2001, pp. 340–345 vol.1.

[29] I. CVX Research, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sep 2012.

[30] J. Lepropre, S. Balon, and G. Leduc, "Totem: A toolbox for traffic engineering methods," Poster and Demo Session of INFOCOM'06, Apr 2006.

[31] D. Xu, "Compact formulation of network entropy maximization," in *Proceedings of the 46th Annual Conference on Information Sciences and Systems (CISS)*, Mar 2012.